

**PHYSICALLY-AWARE DIAGNOSTIC
RESOLUTION ENHANCEMENT FOR DIGITAL
CIRCUITS**

Yang Xue

August 2016

Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:

Ronald D. Blanton

Xin Li

Ying Zhang

Anne E. Gattiker

Jeff Schneider

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.*

Copyright © 2016 Yang Xue

Keywords: Diagnosis, Resolution, Physical failure analysis, Machine learning, Active learning

Abstract

Diagnosis is the first analysis step for uncovering the root cause of failure for a defective chip. It is a fast and non-destructive approach to preliminarily identify and locate possible defects in a failing chip. It is a software-based method that analyzes the applied tests, the failed tester response, and its netlist/layout to produce a list of diagnostic candidates that represent the locations and sometimes the behaviors/types of defects in the chip.

The conventional objective of diagnosis for identifying failure locations has been augmented with various physically-aware diagnosis techniques that are intended to improve both 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. To address this major challenge, a novel, machine-learning-based resolution improvement methodology named PADRE (Physically-Aware Diagnostic Resolution Enhancement) is described.

PADRE uses easily-available tester and simulation data to extract features that uniquely characterize each candidate. PADRE applies machine learning to the features to identify candidates that correspond to the actual failure locations. Through various experiments, PADRE is shown to significantly improve resolution with virtually no negative impact on accuracy. Additional experiments demonstrate that PADRE is robust against data set variation and feature-data availability. Specifically, in simulation experiments, the number of defects that have perfect resolution is increased by $5\times$ with little degradation of accuracy. Multiple silicon experiments also demonstrate similar resolution improvements.

An important investigation that typically follows diagnosis is Physical Failure Analysis (PFA). PFA is a time-consuming and destructive approach for exposing the defect in order to characterize the failure mechanisms to ultimately uncover root cause. PFA relies on an accurate and high-resolution diagnosis to locate the possible defect. In addition, however, PFA can provide information that is helpful for improving diagnosis. PADRE influences PFA within a novel, active

learning (AL) based PFA selection approach. An active-learning based PADRE (AL PADRE) selects the most useful defects for PFA in order to improve diagnostic resolution. AL PADRE uses an alternative defect selection procedure, for improving the overall accuracy and resolution of PADRE. AL PADRE is validated using both simulation and silicon-based experiments. Simulation experiments show AL PADRE can reach an accuracy of 90% with 60% less PFA, on average, compared to conventional defect selection for PFA. In the silicon experiment, by using AL PADRE, the number of chips needed to undergo PFA was reduced by more than $6\times$ in order to increase diagnosis accuracy by more than 20%.

During the yield learning process, the failing mechanisms that lead to defective chips may change due to perturbations in the fabrication process. It is important for PADRE to perform robustly through the entire yield learning process, therefore, additional techniques are developed to monitor the effectiveness of PADRE in real time, as well as to update PADRE efficiently and stably to cope with changing failure mechanisms. It is shown that with these additional techniques, PADRE can robustly perform through various failure mechanisms change even at early stage of the yield learning when there is insufficient diagnosis data for training.

Acknowledgments

Firstly, I would like to thank my supervisors Dr. Zhang Ying, Prof. Xin Li and Prof. Shawn Blanton, for their continuous support and help in this journey. I enjoy working with them, and have great respect and admiration for their insights, dedication and wisdom. I feel truly honored and fortunate to have opportunity to work with them, to know them in person and to learn from them, not only for how to advance in my study and research but also for how to become a better person.

I would like to express my gratitude to Dr. Anne Gattiker and Prof. Jeff Schneider for sitting in my PhD committee. They have shown great patience and understanding to me through many ups and downs. I really want to thank them for taking time from their busy schedules to follow up on my research and to provide many valuable suggestions and feedbacks.

I also want to express my gratitude to A*STAR, Singapore for their financial support, I greatly appreciate their trust on me and their continuous support since even before I embarked on this journey more than six years ago. Without their support, I would never have had this great opportunity to realize my dream and to explore what I love to do.

Last but not least, I cannot express enough how grateful I am to all my friends and family who have supported me through all these years. Some of them help me with my research by answering many stupid questions from me, some of them support me emotionally when I am down, but above all, all of them share their life with me - their happiness, love and sorrow. I want to thank them for allowing me to be part of their amazing life.

Contents

- 1 Introduction 1**
 - 1.1 PADRE 7
 - 1.2 AL PADRE 7
 - 1.3 Changing Failure Mechanisms 8
 - 1.4 Dissertation Organization 9

- 2 Background 11**
 - 2.1 Defects and Faults 11
 - 2.2 Diagnosis 12
 - 2.2.1 Cause-Effect Diagnosis 15
 - 2.2.2 SSL Faults 15
 - 2.2.3 Bridge Faults 16
 - 2.2.4 Transistor Stuck-Open Faults 18
 - 2.3 Per-Test Diagnosis 19
 - 2.4 Summary 21

- 3 PADRE 23**
 - 3.1 The PADRE Methodology 23
 - 3.1.1 Candidate Features 23
 - 3.1.2 Classifier Structure 28

3.1.3	Feature Selection	31
3.2	Experiment with Simulation Data	32
3.2.1	Setup	33
3.2.2	Fisher Score	33
3.2.3	Sequential Forward Selection	34
3.2.4	Resolution Improvement and Accuracy	38
3.2.5	Dataset-Size Sensitivity	39
3.3	Experiments with Silicon Data	41
3.4	Computational Cost	46
3.5	Summary	47
4	AL PADRE	49
4.1	AL PADRE Methodology	49
4.1.1	Discrepancy Check	51
4.1.2	Within-Margin	52
4.1.3	Stopping Criteria	54
4.1.4	Implementation	55
4.2	Simulation Experiments	57
4.2.1	Setup	59
4.2.2	Results	60
4.3	Silicon Experiment	63
4.4	Summary	64
5	Changing Failure Mechanisms	71
5.1	Accuracy Tracking	72
5.2	Changing Failure Mechanisms Training	73
5.3	Cross-Dataset Training	74
5.4	Experiments	75

5.4.1	Setup	75
5.4.2	Accuracy Tracking	77
5.4.3	Changing Failure Mechanism Training	78
5.4.4	Cross-Dataset Training	81
5.5	Summary	83
6	Summary and Future Work	85
6.1	Dissertation Contribution	85
6.2	Future Work	88
	Bibliography	91

List of Figures

- 1.1 The cumulative diagnostic resolution distribution of defects from a commercial chip shows that only 161 or 31.8% of a population of 507 defects have ideal resolution. The plot sorts the defects by the number of candidates for each defect. Each point on the plot shows the number of defects in the entire population that have candidates no greater than a certain number, ranging from one to the maximum number of candidates for a single defect. 4

- 3.1 PADRE takes all the candidates through a two-level classifier, in which incorrect candidates are eliminated and correct candidates are identified. By reducing the total number of candidates while maintaining accuracy, the resolution of the diagnosis is improved. 24

- 3.2 Example of the neighborhood of a candidate associated with net S_6 : (a) the physical neighbors, *i.e.*, nets in close physical proximity of the candidate and (b) the driver and receiving-cell side inputs of the candidate. 26

- 3.3 The diagnosis accuracy improves following the addition of each selected feature. The accuracy plot shows a sharp increase with the addition of the top few features. As more features are selected, the accuracy increase slows and eventually saturates before all the features are selected. 37

3.4	(a) Resolution improvement through PADRE at the defect level shows that the number of defects exhibiting ideal resolution is increased by more than $5\times$. (b) Each defect is plotted according to its improved resolution versus its original resolution. Defects with accurate resolutions are marked with blue circles, whereas defects with inaccurate resolutions are marked with red crosses. The accuracy is maintained for 3,249 or 99.2% of all defects. Bubble size is proportional to the number of defects with the corresponding original/improved resolution.	40
3.5	Resolution improvement for the first silicon data set shows the number of defects that exhibit ideal resolution is increased by 77.6%.	44
3.6	Resolution improvement for the second silicon data set shows that the number of defects exhibiting ideal resolution is increased by nearly $2\times$	46
4.1	Two data sets (crosses and triangles) are plotted with respect to two features, x_1 and x_2 . Solid crosses and triangles are labeled training data from two classes, while unfilled crosses and triangles are unlabeled data. The support vectors (grey crosses and triangles) define the current boundary and its margin. Labeling the unlabeled data within the margin leads to a new, optimal boundary.	53
4.2	PADRE accuracy versus the number of PFA'ed defects selected by AL PADRE. The initial accuracy is 77.0%.	61
4.3	The cumulative diagnostic resolution distribution shows that PADRE improves the number of defects with ideal resolution by more than $5\times$ over the original resolution, and AL PADRE further improves the number by 107. Each point on the plot shows the number of defects that have a total number of candidates no greater than the given cumulative resolution.	62
4.4	PADRE accuracy versus the number of PFA'ed defects selected by (a) random selection and (b) AL PADRE. For random selection, the average accuracy of the 100 experiments is plotted as a bold blue curve. The initial accuracy is 77%. . . .	65

4.5	PADRE accuracy versus the number of PFA'ed defects selected by AL PADRE when defects are selected without the five-candidate restriction.	66
4.6	PADRE accuracy versus the number of weeks of PFA using three different processes, namely, PFA with AL PADRE, PFA with random selection and no PFA. Note that the first four weeks of accumulation is not shown in the plot, as no PFA is performed during that period.	67
4.7	PADRE accuracy versus the number of weeks of PFA using three different processes, namely, PFA with AL PADRE, PFA with random selection and no PFA. Note that there is no accumulation in this setup, all three start from week one. . .	68
4.8	PADRE accuracy versus the number of PFA'ed defects for selection using AL PADRE and actual selection made by industry. The initial accuracy is 61.2%. . .	69
4.9	The cumulative diagnostic resolution distribution shows PADRE with actual PFA'ed defects increases the number of defects with ideal resolution from 88 to 107, and using AL PADRE can instead improves the number to 123. Each point on the plot shows the number of defects that have a total number of candidates no greater than the given cumulative resolution.	70
5.1	The accuracy tracking shows steady increase despite the variance on defect composition (as shown in Table 5.2).	78
5.2	The PADRE accuracy of using a cross-validated weight.	80
5.3	The average accuracy using cross-validated weight consistently outperforms that of using a fixed weight of 20.	81
5.4	The accuracy of using cross-validated weight to combine training data from two different designs.	82
5.5	The average accuracy with combined training data slightly but consistently outperforms that of using training data from a single design.	83

List of Tables

3.1	List of Features Considered in PADRE.	25
3.2	Faults Injected in Failing Chips.	34
3.3	Fisher Scores for Synthetic Data.	35
3.4	Sequential Forward Selection of the Features.	36
3.5	PADRE Candidate-level Classification Statistics.	39
3.6	Dataset Sensitivity Analysis with Reduced Subsets for Training.	42
3.7	Resolution Improvement of PFAed Failing Chips.	45
4.1	Discrepancy Check.	52
4.2	Faults Types Injected into the ASIC Design.	59
5.1	Faults Types Injected into the ASIC Design.	76
5.2	Faults Types of Failed Chip for Each Week.	76
5.3	Cross-validated weight and accuracy.	79

Chapter 1

Introduction

The semiconductor industry has undergone an explosive development in the past few decades, and revolutionized the way people live in the process. The technology node of integrated circuits (IC) has shrunk from $10\mu\text{m}$ in 1971 to 10 nm in 2016, with billions of transistors now packed in a typical chip that powers our smart phones [1]. For decades, the scaling of transistors, driven by Moore's Law, has brought down the cost of semiconductor manufacturing, however, as the technology node advances beyond 10 nm, it becomes evident that further scaling can no longer reduce the cost, as it becomes increasingly difficult to manufacture smaller features. In addition to the manufacturing cost, the testing of the chips also becomes more and more costly, which further shrinks the profit margin of the semiconductor industry [2]. Albeit costly, the testing is an indispensable step in the production. Considering the extremely delicate structure, the complexity of the manufacturing process, and the inherent process variation, it is of no surprise that some manufactured products may be defective. Testing filters out the defective products and safeguards the correct functionality and performance of the shipped products.

Under such circumstance, it is apparent that any defective products that cannot be shipped cut into the profit margin of the production, which is already thin. To ensure the profitability of a product, maintaining a profitable *yield* becomes extremely crucial. Yield is the percentage of manufactured products that function correctly as designed. It is not uncommon for a newly-

introduced design to experience low yield in the early stages of production. It is therefore critical to improve yield as quickly as possible, and a fast yield ramp in turn heavily depends on the ability to accurately and precisely uncover the root-cause of the defective products.

Diagnosis is a fast and non-destructive approach to identify and locate defects in a failing chip [3]. It is a software-based method that analyzes the applied tests, the chip tester response, and the netlist/layout to produce a list of scored diagnosis *candidates* that represent the locations and sometimes behaviors/types of defects within the failing chip. A candidate is different from a defect (*i.e.*, the actual physical deformation that leads to malfunction), in that it is only a representation of the likely location and behavior of a defect.

Many diagnosis techniques identify candidates by comparing the collected tester response of a failing chip with the fault simulation responses for a specific set of deduced faults using the same test patterns, *i.e.*, applied inputs for the test. If an exact match is found, a diagnosis with an ideal resolution is produced. However, an exact match is not common, meaning that there is often mismatch between the simulation response and the tester response. When an output or test pattern that passes on the tester is predicted by the fault model to fail, a mismatch called a Tester Pass Simulation Fail (TPSF) results; similarly, when a fault predicts an output or test pattern to pass which turns out to fail on the tester, it is called a Tester Fail Simulation Pass (TFSP). For matching responses, where an output or test pattern pass or fail both on the tester and in fault simulation, a Tester Pass Simulation Pass (TPSP) or a Tester Fail Simulation Fail (TFSF) results, respectively. To quantitatively evaluate the extent of match and mismatch of various candidates, a ranking method is typically invoked (*e.g.*, in [4]), where a normalized score is assigned to each candidate based on the weighted sum of the matches and mismatches. A diagnosis outcome with few candidates (*i.e.*, good resolution) and very high scores are assumed to correlate with the actual defect(s) that caused failure.

Much effort has been dedicated to improving diagnosis because it is an important part of the yield-learning process for understanding the root cause of failure. Diagnosis can be followed by physical failure analysis (PFA), a time-consuming and destructive approach for exposing the

defect physically in order to characterize the failure mechanisms [5]. 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 failing chips also serve as input for a variety of analyses besides PFA. For instance, volume diagnosis can reveal important statistics including the defect distribution or the primary yield detractors [6, 7], and provide useful feedback for evaluating and improving the quality of manufacturing test [7, 8, 9].

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 set is economically viable, there still can be candidates that have equivalent behavior among the many locations that are specific to the standard cells used and their interconnections. Also, the fault models adopted for both test and diagnosis are not perfect either, meaning it is quite likely that the actual defective behavior cannot be fully explained by the fault model(s) adopted [10]. The overall result is an imperfect diagnosis that typically produces *an accurate result but a non-ideal resolution*. Here, a diagnosis is *accurate* if the true defect location is included in the reported candidate list, whereas the *resolution* is the total number of candidates reported for each defect. The diagnosis outcome is considered *ideal* if and only if the resolution equals to one and it is accurate. However, diagnosis often reports more than one candidate for a defect, where one very likely corresponds to the actual defect whereas the remaining do not. Figure 1.1 illustrates this point by showing the cumulative diagnostic resolution of defects from a commercial chip. It can be seen that just over 30% of the diagnosed defects exhibit ideal resolution.

It is possible to improve resolution with add-on techniques that rely primarily on easily-obtainable data. In particular, resolution improvement can be accomplished through the derivation of characteristics that enable *correct* candidates (candidates that correctly represent defect

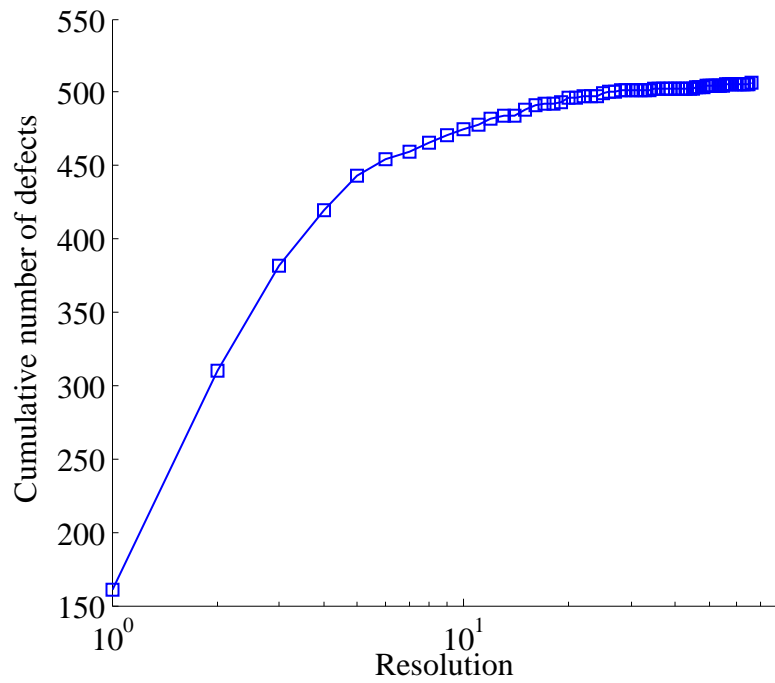


Figure 1.1: The cumulative diagnostic resolution distribution of defects from a commercial chip shows that only 161 or 31.8% of a population of 507 defects have ideal resolution. The plot sorts the defects by the number of candidates for each defect. Each point on the plot shows the number of defects in the entire population that have candidates no greater than a certain number, ranging from one to the maximum number of candidates for a single defect.

locations) to be distinguished from *incorrect* ones (candidates that do not). For instance, the candidate scoring technique mentioned earlier is a proven approach for separating correct and incorrect candidates. For example, in [11, 12], it is suggested that a candidate detected by many tester-passing patterns is less likely to correlate to the actual defect location. Other work reveals that the same neighborhood state (*i.e.*, the logic state of nets near the candidate) for a correct candidate should not be observed for both TPSF and TFSF patterns [13, 14, 15]. If such a situation occurs, the candidate is said to be *inconsistent* and is likely to be incorrect [13, 14, 15]. For example, in [15], it is reported that on average 64% of the incorrect candidates are correctly identified through a check of consistency; and in [14], population of the incorrect candidates are reduced by 62%. Although these techniques are effective in identifying candidates that are likely to be incorrect, they do not directly identify correct candidates. This somewhat limits their overall improvement in resolution, particularly for defects with a large number of candidates. In such cases, even if some of the incorrect candidates are eliminated, without a means to directly identify the correct candidate among the set of all candidates, the resolution may still not be sufficient for practical PFA or other follow-on analyses.

Other than directly improving the diagnosis algorithm itself, other approaches have been reported for improving resolution. Diagnosis-oriented ATPG has drawn considerable work. For example, in [16], the authors introduce a transistor-level and defect-based ATPG for more precise test and diagnosis; in [17], the authors use fault-distinguishing ATPG to create tests that specifically detect a targeted fault without activating other faults. In [18], the authors describe an approach that directly targets faults that are derived from a physically-aware diagnosis. In [18], the correct candidates of 31% of the defects are successfully identified, whereas in [17], 97.4% of the correct candidates are successfully identified. However, the diagnostic APTG typically targets at distinguishing a pair of two candidates at a single time. Therefore, for defects with more than two candidates, the diagnostic ATPG typically involves iterative testing and test pattern generation cycles to incrementally distinguish all the faults, which causes additional test-generation time and cost.

Machine learning (ML) techniques have also been used in diagnosis, especially in various volume diagnosis approaches. In [19], the authors use a two-step approach consisting of rules and a classifier to identify bridge defects from a population of diagnosed failures. In [20], a classifier that takes the tester response as input is used to predict the fanout-free sub-circuit that most likely contains a defect. Other ML-based diagnosis approaches deal with resolution improvement more directly. In [21], the authors use an incremental k -nearest neighbors classifier to improve resolution of an on-chip diagnosis technique meant to identify a failing sub-circuit. In [22], a technique called RCD (root cause deconvolution) uses a Bayesian classifier to determine the defect type responsible for failure, (*e.g.*, opens at certain metal layers, bridges, or cell-level defects). RCD indirectly improves resolution by eliminating candidates that are not associated with the defect type identified by RCD. In [23], the authors describe a technique called DREAMS (DFM rule evaluation using manufactured silicon), where Expectation-Maximization technique is used to identify the DFM (design for manufacturability) rule violation responsible for failure. Similar to RCD, DREAMS improves diagnostic resolution by disregarding candidates that are not associated with the violated rule.

The issue with ML-based techniques mainly lies in the data required to learn the model. For example, in [22], in addition to the conventional test and diagnosis data, detailed layout and fabrication data are also required, which may limit the applicability of the technique. In [20], the classifier is trained with simulation data with injected virtual faults, which raises doubt about the quality of the training data, especially considering the fact that the actual tester responses of defective chips are increasingly difficult to explain with any single fault model [6].

This work attempts to contribute to the effort of improving diagnostic resolution with an easy-to-implement approach that does not require additional testing or physical analysis. We attempt to perform data mining on the easily-available diagnosis data, and try to extract information that are not directly visible in the original data to help improve resolution.

1.1 PADRE

PADRE (Physically-Aware Diagnostic Resolution Enhancement) [24] improves diagnostic resolution through the use of Machine Learning (ML). Specifically, candidate-specific features are derived to characterize and distinguish the diagnosis candidates. Some of the features are well established such as TFSP, TPSE, and TFSF. Some others are new, established in this thesis for characterization of more sophisticated candidate properties. The feature data from a population of candidates are used to learn a classifier that separates correct candidates from incorrect ones.

In PADRE, we employ a two-level classifier. The first level is a simple rule-based check of the neighborhood consistency of each candidate, which is very adept at identifying some types of incorrect candidates [3, 7, 13, 14, 15]. The second level uses an SVM (support vector machine) [25] classifier that is learned from some of the candidates that pass the first level to predict which candidates are correct. PADRE essentially improves the diagnostic resolution through the process of incorrect-candidate elimination and correct-candidate identification.

1.2 AL PADRE

Considering the limited resources for PFA against the vast number of failed chips, the question of which chip, and more specifically, which defect, should be selected for PFA becomes a tricky one. Note that although each failed chip may contain multiple defects, typically only one of the defects can be investigated by PFA, due to its inherently destructive nature. To help identify the most valuable defect for PFA, we describe an active learning-based methodology that can be implemented in conjunction with PADRE to select the most informative defect for resolution improvement. Active learning (AL) has been shown in many other applications to improve the accuracy of classification. It has seen wide application in text classification, image recognition, and early disease diagnosis [26, 27, 28]. AL sequentially queries the label of certain unlabeled data instances for the purpose of improving classification accuracy through improved training

data.. An AL algorithm selects the most informative instance based on pre-set criteria that best suit the application. For diagnostic resolution improvement, however, successful PFA of a defect can reveal labels (correct or incorrect) of multiple instances (*i.e.*, all the reported candidates)¹. To leverage this batch-mode property, we adopt and combine both the conventional AL selection criterion, and a novel selection criterion specifically based on the diagnosis application. The two methods are named *within-margin* and *discrepancy check*, respectively. It is shown through experiments that the two methods are very powerful in improving the accuracy of classification, because they both avoid selecting defects that are already correctly labeled by the existing classifier.

1.3 Changing Failure Mechanisms

As with any machine learning based technique, the performance of PADRE is influenced by the quality of the training data, *i.e.*, whether the training data can properly represent the actual data being analysed. The training data of PADRE are extracted from the diagnosis data, which are the result of certain defects within the failed chips. In the yield-learning process, as the production changes, it is possible that the failure mechanisms may change. It is important that PADRE is able to adapt to the changing failure mechanisms to ensure its accuracy throughout its deployment.

To address this need, an accuracy tracking method is proposed. Accuracy tracking works by applying the existing classifier to some of the latest diagnosis data, of which the labels can be inferred heuristically. Then to update the classifier with the latest diagnosis data, the latest training data is weighted, so that the updated classifier can produce best possible accuracy for the most current diagnosis data.

PADRE extracts training data from the diagnosis data themselves using effective heuristics.

¹Successful PFA of a defect reveals the labels of all candidates associated with a defect because success implies the root-cause has been identified which in most cases the location, and thus the correct candidate has been identified. While one can probably cite rare cases where this would not be true, we ignore such possibilities in this thesis.

Although this approach is shown to be very reliable for constructing high quality training sets, the amount of training data that can be extracted is limited by the total diagnosis data available. When the available diagnosis data are limited, which may happen in the early stage of the yield-learning process, the amount of high-quality training data is consequently limited. A straightforward solution to this situation is to accumulate the diagnosis data until sufficient training data can be extracted, which hinders the yield learning process, as the application of PADRE has to be delayed until sufficient diagnosis data are accumulated.

Another approach to address the lack of data is the use of diagnosis data from other designs. The fact that it is possible to make prediction on a population of diagnosis data by learning a classifier from the diagnosis data of a different design implies that the features used by the classifier are somewhat design-independent. When combining the diagnosis data of two different designs, the key question is how to balance the weight of the data from the two different designs. Intuitively, if there are a sufficiently large amount of diagnosis data available for the targeted design, PADRE should be primarily trained using that. On the other hand, if there is limited diagnosis data from the targeted design, PADRE should rely less on its diagnosis data. Similar to the weighting proposed for handling changing failure mechanisms, weighting is proposed to balance the training data from different designs, so that the PADRE is accurate for predicting candidates from the targeted design.

1.4 Dissertation Organization

The rest of the dissertation is organized as follows: Chapter 2 provides the fundamental for testing and diagnosis; Chapter 3 describes the PADRE technique with a comprehensive study on its resolution improvement capability and analysis of its application under different scenarios; Chapter 4 shows how AL PADRE leverages PFA for significantly increasing the accuracy of PADRE; Chapter 5 describes a number of techniques that help PADRE cope with changing failure mechanisms issues that can occur in the yield learning process; finally Chapter 6 summarizes

major conclusions and contributions of this dissertation, and discusses areas for future work.

Chapter 2

Background

2.1 Defects and Faults

A defect is a physical deformation that changes the circuit structure and/or material properties of the chip. Failing outputs due to a defect result from errors on one or more signal lines upon application of certain test patterns to the Circuit under Test (CUT). Logical fault models are used to capture the logic-level misbehavior of defects. A fault is one instance of a fault model, representing a possible location of the modeled defect. Specifically, a fault represents when and where an error appears upon test application. For example, the single stuck-line (SSL) model, which is the most commonly used fault model, assumes that only one signal line can be permanently stuck at either 0 or 1 [29, 30, 31]. This means, for an SSL fault l/v , where l is one signal line in a CUT, l has an incorrect value v for every test-pattern that drives l to the logic-value v . Similar to the notion of test pattern response, fault-simulation response for a given fault is defined as the output response produced by the CUT in simulation after applying a given test pattern in the presence of the fault. Correspondingly, fault-simulation response is the set of fault simulation pattern responses for the applied test set. A fault is said to *explain* a test pattern t_i if the fault simulation pattern response is identical to the test pattern response for t_i .

Another frequently used fault type is the bridge fault [32]. It models the behavior of unwanted

connections between two or more lines. Depending on the nature of the connection, different types of bridge behaviors have been proposed to capture the resulting misbehavior. The simplest and most commonly used bridge fault models include the AND-type, OR-type and dominating-type, all of which assume a zero-resistance connection between the bridged lines [29]. These bridge faults abstract away the complex electrical characteristics of the defect by assuming that both lines will attain the same logic value. An AND (OR)-type fault assumes that the attained value is 0 (1), while the dominating-type assumes that the logic value of one line will prevail over the others. Other types of complex bridge faults consider a non-zero impedance of the connection for determining fault behavior [33, 34, 35].

Fault models for other types of defects have also been proposed. For example, the transistor stuck open model introduced in [36] assumes that a defect can cause a transistor to be permanently non-conducting or stuck-open. Multiple stuck-line (MSL) [29] faults are used to model the behavior of defects that cause stuck-faults on more than one line simultaneously. Defects that impact the delay of a CUT can often be modeled using transition [37, 38] and path delay faults [39]. The interconnect open fault model presented in [40, 41] is used to model the misbehavior due to resistive (or missing) vias along an interconnect.

2.2 Diagnosis

Fault diagnosis dates back to the mid-1960s when the first algorithms for test generation were proposed [30]. The early diagnosis approaches evolved as a simple outgrowth of automatic test pattern generation (ATPG) algorithms, which use one or more fault models for generating test patterns. When a CUT failed one or more test patterns during manufacturing test, it was only natural to assume that the fault(s) associated with the failing patterns must contain some attributes of the defect in the CUT. In other words, because the failing patterns (that detect the defect in the CUT) were generated targeting some fault during the ATPG process, the same fault must somehow capture some aspect of the actual defect. Starting from this simple and intuitive

observation, fault diagnosis has become a necessarily complex task.

The need for more powerful fault diagnosis techniques stands in contrast to the needs of ATPG. Although simple fault models (typically, SSL or transition faults) suffice for generating test patterns that detect defects in nanoscale ICs, fault diagnosis based solely on the same fault models are rarely able to identify a fault that accurately represents the defect in the CUT. Some fault diagnosis approaches attempt to improve accuracy by adopting more complex fault models [10, 35, 42, 43], others use simple fault models in conjunction with complex matching techniques for localization [11, 44, 45]. Yet others focus exclusively on diagnosis of defects that behave as multiple stuck-line (MSL) faults [46, 47, 48]. In complimentary work, some researchers focus on fault-distinguishing test pattern generation for improving accuracy and resolution of existing diagnosis techniques.g., [17, 49].

Early on, fault diagnosis techniques were categorized as either localization or fault-identification methods based on their end objective of identifying defect locations or faults, respectively. Fault diagnosis techniques can also be classified based on how they achieve their respective objectives, i.e., based on their approach. Specifically, all fault diagnosis techniques can be very broadly categorized into either cause-effect or effect-cause analysis techniques. The former identifies candidates by comparing the observed tester response of a CUT with fault simulation responses for a set of faults chosen a priori. Therefore, cause-effect techniques start with possible causes (faults) and determine if any selected *cause* can account for the *effect*, i.e., the tester response.

Similar to human diagnosticians in its main principle of relying on known failure modes, the cause-effect category of techniques had been the most popular. The fault simulation responses utilized in cause-effect techniques are typically generated once and stored in a simple but large database typically referred to as a fault dictionary [29, 30, 50, 51]. Fault dictionaries suffer from both space and time complexities, that is, the compute time and memory required to fault simulate and store fault-simulation responses for all the targeted faults is extremely prohibitive. Although research in dictionary compaction techniques have attempted to alleviate the storage issue, e.g., [52], the dictionary approach is still limited by long simulation time and its inher-

ent inefficiency in that only a small fraction of the fault simulation responses are ever used for diagnosing failures.

The success of cause-effect techniques hinges upon the availability of accurate fault models. If a fault model precisely captures the behavior of a defective CUT, a simple (exact) matching criterion can be used for identifying diagnosis candidates. Given that fault models seldom accurately model defects [12], fault diagnosis research has focussed on developing complex response matching techniques and an alternative approach known as effect-cause analysis. Rather than searching for a traditional fault whose simulation response is identical to the tester response, diagnosis approaches based on matching techniques utilize the notion of temporary stuck line (TSL) faults and a ranking metric to identify sets of TSL faults that together explain all the failing patterns. The signal lines associated with the identified set of TSL faults are reported as candidate locations.

Effect-cause analysis techniques do not use fault dictionaries but instead begin with the effect in order to find its potential causes [29]. Specifically, effect-cause analysis starts from the failing outputs and reasons back through the CUT to identify lines that (if faulty) could have caused the failing outputs. Most effect-cause techniques begin with some form of critical path tracing [53]. Although effect-cause techniques claim to be completely independent of fault models, they employ rules of fault sensitization and error propagation that are similar to SSL faults. In other words, effect-cause analysis techniques, at least in their earliest form [29], attempt to identify signal lines that cannot be driven to both logic-0 and logic-1 for the applied test patterns, which is the understood definition of an SSL fault.

A typical diagnosis output of effect-cause analysis consists of a set of signal lines or logical regions of the CUT that are suspected to be defective. The fact that effect-cause techniques can identify fault locations even when multiple lines are faulty simultaneously is another advantage over cause-effect techniques that are typically based upon a single-fault assumption [29, 51]. On the negative side, effect-cause analysis focusses on localization and does not characterize the behavior of the failure. In addition, because effect-cause techniques tend to be conservative and

include many of the signal lines that could have caused the observed failing outputs, diagnosis resolution is typically low.

Since an IC can fail in a vast number of ways, the model-independent approach of effect-cause techniques is more practical than the cause-effect approach based on one or more fault models. However, use of accurate fault models increases diagnosis accuracy and resolution [10, 35]. Also, fault models enable characterization of defect behavior. In order to get the best of both worlds, most fault diagnosis techniques use a combination of effect-cause and cause-effect techniques [11, 14, 54, 55]. Specifically, potential defect locations are first identified using effect-cause analysis and fault models are then applied based on the reduced set of signal lines to identify candidates via a dynamic cause-effect analysis.

2.2.1 Cause-Effect Diagnosis

The main objective of diagnosis based on cause-effect analysis is to identify a single fault (based on a presumed fault model) that can predict the observed CUT behavior. While initial techniques used SSL faults, bridge [10, 35] and transistor stuck-open faults [56] have also been utilized. In this section, we will review some cause-effect diagnosis techniques based on the fault models utilized.

2.2.2 SSL Faults

The simplicity, success, and availability of existing ATPG tools for the SSL fault model made it a natural choice for fault diagnosis [30]. The earliest diagnosis techniques searched for SSL faults with simulation responses identical to the tester response. However, experience quickly revealed that stuck-at faults did not always accurately model the defect behavior observed from failing CUTs. As a result, more complex response comparison heuristics were developed to identify candidate SSL faults. The notion of fuzzy matching was used by Western Electric Company as early as 1971 to identify SSL faults whose fault-simulation responses approximately matched

the tester response [57].

Because an SSL fault does not accurately capture the behavior of many complex defect types, mispredictions and nonpredictions can occur. A misprediction occurs when a fault predicts an output to fail that does not fail on the tester, while the absence of one or more observed failing outputs in the fault simulation response results in a nonprediction. To deal more effectively with mispredictions and nonpredictions, a new method for ranking SSL faults was introduced in [4], where a score is assigned to each SSL fault. The score of an SSL fault s_i is calculated as a weighted sum (the weights being user-defined inputs) of the following two metrics: 1) the number of failing outputs in the tester response not present in the fault simulation response, and 2) the summation of the number of passing patterns that detect s_i and number of failing patterns for which s_i does not explain any failing output. The two metrics are zero when intersection equals the set of failing outputs in the tester response and when there are no mispredictions and nonpredictions, respectively. Faults that have the lowest score are the best candidates.

Although comparison heuristics are able to accommodate defects that do not behave exactly like SSL faults, diagnosis based on SSL faults is still limited because of the assumption that defects cause only one signal line in the CUT to become faulty. In addition, the line is assumed to be stuck-at the faulty value permanently. In reality however, there often are defects that cause errors of both polarities (0/1 and 1/0) on lines that can change for different test patterns. In other words, defects can exhibit different misbehaviors for each failing pattern. Diagnosis of such defects based solely on the SSL fault assumption may fail to identify defect locations accurately or may result in an empty diagnosis, i.e., no candidates are identified.

2.2.3 Bridge Faults

Unwanted electrical connections between two lines have long been considered an important class of defects. In the logic-level domain of ATPG, fault simulation and fault diagnosis, the behavior of a two-line unwanted connection is modeled using two-line bridge faults of different types.

Cause-effect diagnosis techniques based on bridge faults utilize one or more bridge fault models to identify candidates.

A dictionary-based approach is described in [35], where diagnosis candidates are identified by simulating bridge faults and finding the faults with simulation responses identical to the tester response. A more accurate model for bridge faults, namely, the biased-voting model is used in [42] and [33]. Very good diagnosis results in terms of accuracy and resolution are reported. Based on the results of their experiments and comparisons with an alternative approach to bridge fault diagnosis, the authors observed that a more generalized fault model can lead to a more accurate fault diagnosis. Although accuracy is improved, their technique suffers from the same limitations as any other cause-effect technique. Specifically, the number of two-line bridge faults is enormous at for an n-line CUT. In [10, 35], layout analysis is used to identify the likely bridge locations, but the number of likely bridges can still be quite large and the time required to simulate and store the fault simulation responses can be prohibitive. Additionally, time-consuming transistor-level analysis is required to build the biased-voting model, but this cost can be incurred once for a standard library and then amortized over many designs.

An alternative cause-effect technique for identifying bridge fault candidates is described in [58]. Rather than using fault dictionaries, these techniques analyze each failing pattern to identify pairs of lines that satisfy the detection conditions for bridge faults. A bridge fault between two lines is detected if the two lines are driven to opposite logical values and a stuck-at fault on one of the two lines is detected. For each failing pattern, the diagnosis technique of [58] searches for pairs of lines that are 1) driven to opposite values for each failing pattern and 2) have a stuck-at fault on at least one of them explain the failing pattern. Because such technique does not use fault dictionaries generated in a static manner, it is sometimes referred to as dynamic cause-effect technique.

The most significant drawback of dynamic cause-effect technique as described above is that its applicability is limited to only those bridge faults that behave like one of the traditional bridge fault types, i.e., AND, OR and dominating bridge faults. In other words, the technique will fail to

identify the correct candidates if the bridge behaves any differently from the traditional models (e.g., if the bridge behaves according to the biased-voting model).

2.2.4 Transistor Stuck-Open Faults

Transistor stuck-open (TSO) faults [36] represent the logic-level behavior of defects that cause a transistor to become non-conducting. Unlike the other fault models discussed so far, TSO faults capture misbehavior stemming from within a gate. As described in [36], SSL faults dominate TSO faults, that is, every test pattern that detects a TSO fault also detects at least one SSL fault at the input or output of the affected gate. Further, TSO faults are assumed to be detected by a sequence of patterns applied to the defective transistor. TSO faults are therefore often referred to as sequence-dependent faults.

A technique to identify gates with TSO faults is described in [59]. This technique is similar in principle to the dynamic cause-effect techniques for bridge fault diagnosis in that it utilizes the detection requirements of an assumed fault type to identify diagnosis candidates. Specifically, gates affected by TSO faults are identified using a two stage process. Initially, potential locations are identified by finding SSL faults that explain all the failing patterns. Given that this technique ignores the behavior of the intragate defect for passing patterns, this technique actually searches for a temporary stuck-line (TSL) fault $(li/v)Tk$, where Tk is the set of failing patterns. A sequence-dependence check is then performed for each potential location; for each TSL fault $(li/v)Tk$ identified in the first stage, the diagnosis technique checks for a transition from logic- v to logic- v for two consecutive patterns. Locations that satisfy the transition check are reported as diagnosis candidates.

The greatest advantage of the above technique lies in its simplicity. As described in [59], this technique can be implemented with only slight modifications to existing SSL fault simulators. However, the technique is not complete in that it is not guaranteed to find the faulty gate since only a detection condition is used and not a model of all possible TSO behaviors.

An alternative TSO fault diagnosis technique based on the same principle as [59] is described in [56]. This technique employs circuit modification, where the transistor-level TSO fault is mapped to an SSL fault by transforming the transistor-level description of a suspect gate into a gate-level description.

Suspect gates are initially identified in terms of TSL faults as described earlier. SSL-based fault diagnosis (with an exact match condition) is then used to identify candidate SSL faults that are subsequently correlated back to the relevant TSO faults. Although this technique allows using the relatively straightforward SSL-based diagnosis, it has the same limitations as [56]. Also, this technique requires transforming the transistor-level description of each suspect gate into its equivalent gate-level description for each run of diagnosis.

2.3 Per-Test Diagnosis

As described in the previous section, cause-effect diagnosis is limited by the lack of accurate fault models as well as the impracticality of simulating and storing the fault simulation responses for non-SSL faults. Also, cause-effect diagnosis techniques lack generality in that their applicability is restricted to the targeted set of fault models. An alternative approach to cause-effect diagnosis involves using advanced matching techniques in conjunction with SSL fault simulation to perform fault diagnosis. This approach is based on the key observation that behavior of most defects match a set of TSL faults. The basic objective of matching techniques is to identify sets of TSL faults that can together explain the observed tester response. A different method for achieving this objective have been described in the past, which is named per-test diagnosis [11]. In this approach, each failing pattern is examined as an independent source of diagnosis, hence the name *per-test*.

The earliest per-test diagnosis technique is described in [11], where SSL faults are simulated iteratively to identify sets of TSL faults that can account for all the applied test patterns. A dynamic cause-effect approach is adopted, where the SSL faults analyzed are selected using

effect-cause analysis for each failing pattern. SSL faults that explain a given failing pattern are fault simulated against other failing patterns to determine if they can be explained as well. If an SSL fault that explains all the test patterns (both failing and passing) is found, it is reported as the candidate and diagnosis stops. Otherwise, the explained failing patterns are removed from further consideration and the corresponding TSL faults are added to the candidate list. Diagnosis is repeated until all failing patterns have been explained, adding one or more TSL faults to the candidate set at each iteration step.

Based on the identified set of TSL faults, the type of defect is binned into one of two classes: 1) defects that exhibit stuck-0 and stuck-1 behavior on the same line (i.e., candidate set has $(l_i/0)T_p$ and $(l_i/0)T_q$, where T_p+T_q equals set of all failing patterns), and 2) defects that exhibit stuck-at behavior on more than one signal line (i.e., several TSL faults in the candidate set). For these two defect classes, some amount of misprediction is possible for the passing patterns, but each failing pattern is expected to be accounted for by at least one TSL fault in the candidate set. Thus, defects that create a failing pattern that is not part of any TSL fault defeat this approach.

More recent per-test diagnosis techniques are described in [6, 12, 60]. The SLAT technique of [6, 60] identifies faulty lines by analyzing each failing pattern independently in a manner similar to [11]. A line l_i is considered faulty if the TSL fault of either polarity on l_i explains at least one failing pattern. The failing patterns that are not part of any TSL fault are discarded; the remaining (explained) failing patterns are referred to as SLAT patterns. The lines obtained from diagnosis of each failing pattern are analyzed using a set-covering algorithm to identify a minimal set that can account for all the SLAT patterns. The minimal set forms the diagnosis outcome and is the representation of the potential defect locations.

In [11], the SLAT philosophy is extended by 1) identifying sets of TSL faults (as against only faulty lines in SLAT) and 2) using probability theory as a basis for ranking the sets of TSL faults. All test patterns are used for determining the rank of a TSL fault set. The TSL faults in the top-ranking set are subsequently correlated with common fault models that include bridge, interconnect-open and other fault types. In the fault model correlation process, inherent

characteristics of different fault types are utilized. For example, an AND(OR)-type bridge fault is inferred if the candidate set comprises only two TSL faults, each with the 0(1) polarity and an interconnect open fault is inferred if the set consists of TSL faults of both polarities on the same line. Therefore, the per-test approach of [11] integrates the model-independent benefits of per-test diagnosis and fault-identification benefits of cause-effect diagnosis.

2.4 Summary

In this chapter, some background about testing and diagnosis are provided. Defects are physical deformations that change the circuit structure; faults are the models used to capture the logic-level misbehavior of defects. Diagnosis is the process to localize and identify the faults in defective chips.

Diagnosis techniques are categorized as either cause-effect or effect-cause. Cause-effect techniques require accurate fault models, from the most basic SSL fault model to other more sophisticated fault models, an accurate cause-effect diagnosis not only localizes the defect, but also reveals the nature of the defect. On the other hand, effect-cause techniques, such as per-test diagnosis, do not rely on specific fault model, which makes them more versatile for diagnosing the defects that cannot be accurately modeled with fault models.

However, despite all the efforts, diagnosis is still not perfect. The poor diagnostic resolution hinders other analyses for yield learning.

Chapter 3

PADRE

3.1 The PADRE Methodology

PADRE is a two-level classifier that identifies incorrect candidates in the first level and correct candidates in the second. The overall flow of PADRE is illustrated in Figure 3.1. PADRE takes as input the diagnosis results for a population of failing chips. Each chip may have multiple defects and each defect may have multiple candidates. For all the candidates, a set of features are extracted to characterize each candidate.

3.1.1 Candidate Features

Candidate features are specific design and testing characteristics that differentiate correct candidates from incorrect ones. The candidate features now considered in PADRE are summarized in Table 3.1.

Some of the candidate features such as `passing_states`, `failing_states`, and `inconsist_states` are physical features, because they characterize the layout characteristics of a candidate when it is both activated and sensitized. Specifically, the neighborhood state of a candidate is formally defined as the logic values driven on nets that are in physical proximity of the candidate for

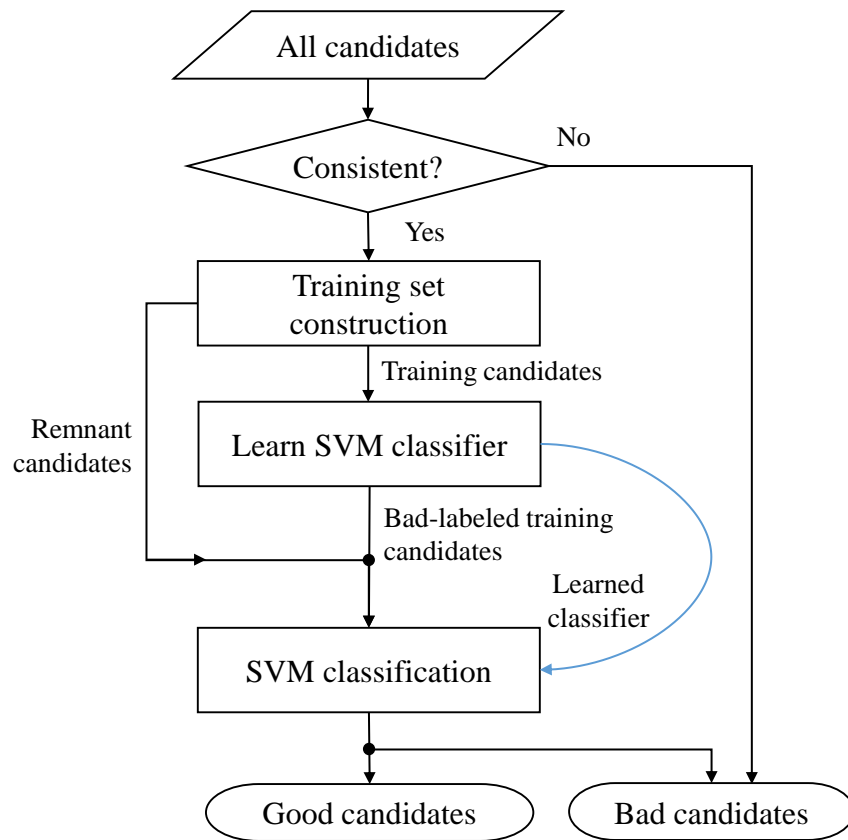


Figure 3.1: PADRE takes all the candidates through a two-level classifier, in which incorrect candidates are eliminated and correct candidates are identified. By reducing the total number of candidates while maintaining accuracy, the resolution of the diagnosis is improved.

Table 3.1: List of Features Considered in PADRE.

Feature	Description
TFSF	No. of Tester-Fail-Simulation-Fail (TFSF) outputs associated with the candidate
TFSP	No. of Tester-Fail-Simulation-Pass (TFSP) outputs associated with the candidate
TPSF	No. of Tester-Pass-Simulation-Fail (TPSF) outputs associated with the candidate
bit_score	Candidate score based on output features
TFSF_ptn	No. of Tester-Fail-Simulation-Fail (TFSF) patterns associated with the candidate
TFSP_ptn	No. of Tester-Fail-Simulation-Pass (TFSP) patterns associated with the candidate
TPSF_ptn	No. of Tester-Pass-Simulation-Fail (TPSF) patterns associated with the candidate
ptn_score	Candidate score based on pattern features
nbr	No. of neighbors of the candidate
failing_states	No. of unique neighborhood states observed for TFSF patterns
passing_states	No. of unique neighborhood states observed for TPSF patterns
inconsist_states	No. of unique neighborhood states observed for both TPSF and TFSF patterns
resol_ratio	The resolution of the defect that the candidate is associated with
ratio_failing_states_to_TFSF_ptn	Ratio between no. of unique failing states and no. of TFSF patterns
ratio_passing_states_to_2 ^{nbr}	Ratio between no. of unique passing states and 2 ^{nbr}
ratio_failing_states_to_2 ^{nbr}	Ratio between no. of unique failing states and 2 ^{nbr}
ratio_max_0_TPSF_physical_to_physical_nbr	Ratio between maximum no. of 0s in TPSF physical states and the no. of physical neighbors
ratio_max_1_TPSF_physical_to_physical_nbr	Ratio between maximum no. of 1s in TPSF physical states and the no. of physical neighbors
ratio_max_0_TFSF_physical_to_physical_nbr	Ratio between maximum no. of 0s in TFSF physical states and the no. of physical neighbors
ratio_max_1_TFSF_physical_to_physical_nbr	Ratio between maximum no. of 1s in TFSF physical states and the no. of physical neighbors
ratio_max_0_TPSF_nbr_to_all_nbr	Ratio between maximum no. of 0s in TPSF states and the total no. of neighbors
ratio_max_1_TPSF_nbr_to_all_nbr	Ratio between maximum no. of 1s in TPSF states and the total no. of neighbors
ratio_max_0_TFSF_nbr_to_all_nbr	Ratio between maximum no. of 0s in TFSF states and the total no. of neighbors
ratio_max_1_TFSF_nbr_to_all_nbr	Ratio between maximum no. of 1s in TFSF states and the total no. of neighbors

tests that detect the candidate [3]. The neighborhood nets for a given candidate, as illustrated in Figure 3.2, include:

- **Physical neighbors** : nets that are in close proximity
- **Drivers** : inputs of the cell that drives the candidate
- **Side inputs** : side inputs of the cells driven by the candidate

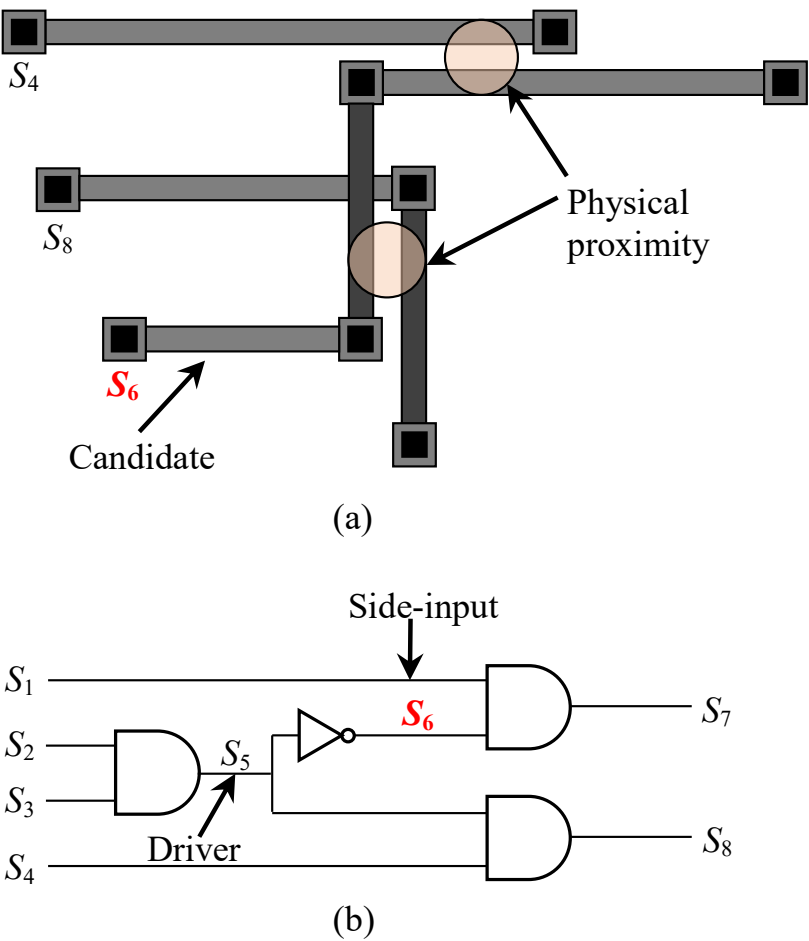


Figure 3.2: Example of the neighborhood of a candidate associated with net S_6 : (a) the physical neighbors, *i.e.*, nets in close physical proximity of the candidate and (b) the driver and receiving-cell side inputs of the candidate.

Independent of any other characteristics of the circuit, the logic value of a candidate, whether faulty or fault-free, is assumed to be a function of its neighborhood state, *i.e.*, the logic values of its neighbors. 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 activation behavior should be a consistent function of its failing and passing states.

Many features used by conventional diagnosis are included in this thesis, such as TFSF, TFSP, and TPSF, which compare/contrast the pass-fail status of the *test outputs* observed by the tester and predicted by simulation, as well as TFSF_ptn, TFSP_ptn, and TPSF_ptn, which compare/contrast the pass-fail status of the *test patterns* observed by the tester and predicted by simulation. In addition to their absolute values, the ratio between the features is also commonly used in characterizing a candidate. For example, in [61] the following ratio-based score is used:

$$\text{bit_score} = \frac{\text{TFSF}}{\text{TFSF} + \text{TPSF}/10 + \text{TFSP}} \quad (3.1)$$

It should be noted that TPSP is not used in the formulation since many candidates typically have very large values for this feature that would overwhelm the significance of the remaining three. Also, TPSF is weighted by 1/10 to cope with the possible temporal stuck-at behavior of a defect, which may produce a large value for TPSF that overwhelms the other parameters in the equation, *i.e.*, TFSF and TFSP.

Similar to *bit_score*, a score based on the ratio between pattern features is also included:

$$\text{ptn_score} = \frac{\text{TFSF_ptn}}{\text{TFSF_ptn} + \text{TPSF_ptn}/10 + \text{TFSP_ptn}} \quad (3.2)$$

In addition to these well-established features, many other ratio-based features are newly established in this thesis. For example, *ratio_failing_states_to_TFSF_ptn* characterizes the variety of different failing states observed for TFSF patterns; *ratio_passing_states_to_2^{nbr}* and *ratio_failing_states_to_2^{nbr}* characterize the variety of different passing states and failing states respectively, given the maximum number of possible different states of 2^{nbr}. Some ratio-based features in this thesis characterize the maximum density of 0s or 1s in the neighborhood states observed for TFSF or TPSF patterns, whereas some other features characterize similar properties

within only physical neighbors. All these features provide information to characterize candidates in different passing/failing situations that are not exploited by conventional diagnosis.

3.1.2 Classifier Structure

PADRE is a two-level classifier; the first-level classifier consists of a single-rule discriminator eliminating incorrect candidates. The second-level classifier is an SVM classifier used to identify correct candidates.

First-level classifier

The first level is a one-rule discriminator that is based on the `inconsist_states` feature. As described earlier, this feature counts the number of unique neighborhood states that are observed for at least one TPSF pattern and one TFSF pattern. The existence of an inconsistent state is likely an indication that the candidate is an incorrect candidate. Any candidate with a non-zero `inconsist_states` is *labeled* as incorrect and is eliminated from the candidate list. A *label* refers to a prediction made by the classifier on whether a given candidate is correct or incorrect. For all the remaining candidates with no inconsistent state, their labels remain unknown and are passed on to the second level. The action of the first-level classifier can be summarized as:

$$\text{label} = \begin{cases} \text{incorrect} & \text{inconsist_states} > 0 \\ \text{unknown} & \text{inconsist_states} = 0 \end{cases} \quad (3.3)$$

Second-level classifier

Although the first level is able to accurately identify a large number of incorrect candidates, typically many candidates still remain unlabeled after the first level. We introduce the second level to further process the remaining unlabeled candidates, with a particular focus on identifying correct candidates. All unlabeled candidates from the first-level classifier are processed by an

SVM-based second-level classifier. The second-level classifier predicts all unlabeled candidates as either correct or incorrect. Based on the correct candidates identified by the second-level classifier, resolution improvement is performed across all the defects. If a defect has any of its candidates predicted as correct by the second-level classifier, then all its remaining candidates which are predicted as incorrect are eliminated from the candidate list. On the other hand, if no candidate is predicted as correct, all candidates associated with the defect are retained in the candidate list.

SVM requires training data to learn a classifier. Unlike conventional supervised learning techniques, where the training data are assumed to be labeled using techniques, such as PFA, that is costly or difficult to invoke, the training data used to learn the second-level classifier in PADRE is derived from the pool of unlabeled candidates themselves. Specifically, the correct-labeled training set is obtained from all the defects with a single candidate. Assuming that diagnosis is accurate, collecting a statistically-significant set of single-candidate defects ensures that the characteristics of correct candidates are well represented by correct-labeled training set. On the other hand, the incorrect-labeled training set is obtained from all the defects with more than Q candidates, where Q is a user-defined value. The heuristic behind the incorrect-labeled training set construction is that there is likely only one correct candidate associated with a given defect. Thus for a defect with multiple candidates, all but one will be incorrect. In general, a larger Q lowers the error, but a smaller value for Q increases the amount of incorrect-labeled training data. By taking all the defects with a high number of candidates, the incorrect-labeled training set will mostly consist of incorrect candidates. For example, in this thesis we choose $Q = 20$, which bounds the error at 5%, in other words, at most 5% of the candidates in the incorrect-labeled training set will be actually correct candidates. Note that the value of Q is chosen based on the property of available failing data, it should be high enough to ensure the quality of labeled-incorrect training data, whereas it should not be too high as that will fail to produce sufficient amount of incorrect-labeled training data to match the size of the correct-labeled training data.

By studying the nature of the candidates in the correct-labeled training set, it is revealed that

the correct-labeled training set consists of only candidates that do not have any equivalent faults, *i.e.*, faults that are indistinguishable from a given candidate by contrasting their test responses. Based on their net types, such candidates can be categorized as follows:

- PI (primary input)
- PO (primary output)
- fanout of PI
- fanout of stem-net
- stem-net with a fault that is not equivalent to any faults at its driver

It should be noted that the correct-labeled training set and the incorrect-labeled training set are likely to be different in size (there are typically fewer correct candidates than incorrect candidates identified from the unlabeled pool of candidates). The imbalance of training set size may result in a biased classifier. Specifically, if the size of the incorrect-labeled training set overwhelms the size of the correct-labeled training set, the learned classifier will be heavily biased towards incorrect candidates, causing it more likely to incorrectly predict an actually correct candidate as incorrect [62]. A straightforward solution for this problem involves balancing the size of the training sets with simple re-sampling techniques. There are two approaches to perform the sampling. One approach is to under-sample the incorrect-labeled training set, and the other approach is to over-sample the correct-labeled training set. Under-sampling the incorrect-labeled training set has the benefit of low computational cost, but it incurs the risk of not properly characterizing the incorrect candidates if the difference between the sizes of two training sets is too large. We therefore choose to balance the two training sets by over-sampling the correct-labeled training set. Specifically, candidates in the correct-labeled training set are randomly duplicated until two training sets are equal in size.

In summary, the construction of the two training sets is performed as follows:

- **Initial training sets:** The initial training set of candidates that are labeled correct consists of all the candidates of defects with only a single candidate; and the initial training set

of candidates that are labeled incorrect consists of all the candidates of defects with more than Q candidates, where Q is a user-defined threshold. In this work, we set $Q = 20$.

- **Balanced training sets:** The smaller initial training set of one class is over-sampled to match the size of the larger initial training set of the other class.

3.1.3 Feature Selection

In practice, some features proposed in this thesis may not be available, which could potentially hinder the performance of PADRE. Therefore, it is important to evaluate how useful each feature is, so as to understand their impact on PADRE performance when they are not available. In addition, it is always desirable to perform resolution improvement at a lower cost. It is possible to reduce the cost of feature extraction if we can reduce the number of features used by PADRE while maintaining classification accuracy. To address these concerns, all the features are analysed to understand their effectiveness within PADRE.

Fisher score [63] is a straightforward measurement for evaluating the correlation between a feature and the candidate class. A higher Fisher score indicates a stronger correlation, and thus a more effective feature. To calculate Fisher score, it requires not only the feature data of the candidates, but also the true labels of the candidates. For feature data \mathbf{x} of n candidates, the Fisher score of j -th feature of the data, $\mathbf{x}^j \in \mathbb{R}^{1 \times n}$ is calculated as:

$$F(\mathbf{x}^j) = \frac{\sum_{k=1}^c n_k (\mu_k^j - \mu^j)^2}{\sum_{k=1}^c n_k (\sigma_k^j)^2} \quad (3.4)$$

where c is the total number of classes corresponding to j -th feature, n_k is the total number of candidates in each class k , μ_k^j and σ_k^j are the mean and the standard deviation of data in the k -th class corresponding to j -th feature, respectively, and μ^j is the mean of all the data corresponding to j -th feature.

Although the Fisher score captures how strongly each feature correlates with the candidate class, it is inadequate to infer the joint effectiveness of multiple features as it does not account

for correlation among features. To evaluate the joint effectiveness of multiple features, a feature selection problem has to be solved.

Given a feature set \mathbf{Y} , where there are n features, the target is to identify a feature subset \mathbf{X} , $\mathbf{X} \subseteq \mathbf{Y}$, based on a criterion function $J(\mathbf{X})$. One common choice for $J(\bullet)$ is $(1 - p_e)$, where p_e is the classification error probability. The selected subset \mathbf{X} should maximize $J(\bullet)$, in order to achieve the lowest classification error. Formally, the feature selection problem can be formulated as:

$$J(\mathbf{X}) = \max_{\mathbf{Z} \subseteq \mathbf{Y}, |\mathbf{Z}|=d} J(\mathbf{Z}) \quad (3.5)$$

where d is the total number of features in \mathbf{X} , *i.e.*, $|\mathbf{X}| = d$.

The procedure to identify the optimal subset that maximizes $J(\bullet)$ can be very costly depending on the total number of features [64], because exhaustively evaluating every subset of d features out of total n features requires $\binom{n}{d}$ rounds of evaluations, which grows exponentially with n . Unfortunately, it has been shown that the optimal subset can only be guaranteed through an exhaustive-selection procedure [64].

Other methods such as Sequential Forward Selection (SFS) may be used to perform the feature selection task in a more computational-efficiently manner. SFS is a greedy, iterative method to identify features that maximize $J(\bullet)$ [65]. It begins with a null feature set, then in every subsequent round, a single feature is added that improves $J(\bullet)$ by the most. Features are continually added one at a time, until no more improvement in $J(\bullet)$ can be made, or the preset number of features d is reached.

3.2 Experiment with Simulation Data

A comprehensive simulation experiment is performed using virtual data to evaluate the performance of PADRE. In the experiment, a substantial number of faults of various types are injected into an industrial design to emulate real failing chips. An injected fault is meant to mimic the

actual behavior of a real defect. The virtual failing chips are tested and diagnosed with typical testing and diagnosis procedures to produce initial diagnosis results. PADRE is subsequently applied to improve the diagnostic resolution, which is then evaluated for the level of improvement and accuracy achieved.

An advantage of using virtual data for an experiment is that all the types and locations of the injected faults are known, which makes it easy for us to identify the true labels of all the diagnosis candidates. Knowing the true labels of the candidates enables us to evaluate the accuracy of resolution improvement, a defect is accurate if the candidate representing the true defect location remains in the candidate list after resolution improvement.

3.2.1 Setup

The chip used for the experiment is an ASIC manufactured in 130 nm technology that contains approximately one million gates. More than 1,600 virtual failing chips are created, each injected with at least one fault. The chips are tested using the production test set that consists of 3,403 logic tests that achieve 99.5% SSL fault coverage. The test results (*i.e.*, fault-simulation responses) are diagnosed using a commercial diagnosis tool to produce a set of candidates.

To ensure that the synthetic data set correctly represents the large variety of defects that could occur in real failing chips, multiple types of faults are injected. The injected faults include: and, or, and four-way bridge faults, input pattern faults[66], SSL faults, and MSL faults. The selected fault types can also mimic the behavior of some other types, *e.g.*, the four-way bridge fault overlaps with the stuck-at-0/1 behavior exhibited by an open fault. The numbers of chips injected for each type of fault are shown in Table 5.1.

3.2.2 Fisher Score

The Fisher score is calculated for each feature used by the second-level classifier, that is, all the features shown in Table 3.1 except `inconsist_states`. The feature `inconsist_states` is not included

Table 3.2: Faults Injected in Failing Chips.

Fault type	No. of chips
AND bridge	96
OR bridge	94
Four-way bridge	275
Input pattern	412
SSL	364
MSL	365

in the feature selection experiment because it is an imperative, and the only feature used by the first-level classifier. In addition, it does not work with other features in any way in the second-level classifier, thus the question of whether it makes up an optimal subset among other features is not applicable.

It can be seen from Table 3.3 that the Fisher score varies significantly from feature to feature, indicating that the importance of each feature also varies significantly. For example, `ptn_score`, `TFSP_ptn`, and `bit_score` have the highest scores, meaning they have strongest correlations with the candidate class. On the other hand, some other features, such as `ratio_max_1_TFSF_nbr_to_all_nbr`, and `ratio_max_0_TFSF_nbr_to_all_nbr`, only have very weak correlations with the candidate class.

3.2.3 Sequential Forward Selection

In addition to Fisher scores, Sequential forward selection (SFS) is also performed as another feature selection experiment. SFS identifies the most effective subset of features by iteratively including the most effective feature one by one to maximize the classification accuracy. It is a correct complement to the Fisher scores which evaluate the effectiveness of each individual feature. The sequence of selected features is shown in Table 3.4, and the diagnosis accuracy following the addition of each feature is shown in Figure 3.3. It can be seen that the top features according to SFS are `ptn_score`, `TFSP_ptn`, and `bit_score`, which are consistent with the conclusion based on the Fisher scores. The top three features alone jointly achieve a diagnosis accuracy close to 90%. However, the accuracy increment slows notably with the addition of more features.

Table 3.3: Fisher Scores for Synthetic Data.

Feature	Score
ptn_score	2.255
TFSP_ptn	1.261
bit_score	0.824
resol_ratio	0.261
ratio_failing_states_to_TFSF_ptn	0.254
ratio_max_0_TPSF_physical_to_physical_nbr	0.203
ratio_max_1_TPSF_physical_to_physical_nbr	0.16
ratio_max_0_TPSF_nbr_to_all_nbr	0.142
ratio_max_1_TPSF_nbr_to_all_nbr	0.137
TFSP	0.116
ratio_passing_states_to_2 ^{nbr}	0.106
TFSF_ptn	0.099
ratio_failing_states_to_2 ^{nbr}	0.082
TPSF	0.04
TPSF_ptn	0.039
ratio_max_0_TFSF_physical_to_physical_nbr	0.036
ratio_max_1_TFSF_physical_to_physical_nbr	0.032
TFSF	0.027
nbr	0.016
passing_states	2.64×10^{-3}
failing_states	2.08×10^{-3}
ratio_max_0_TFSF_nbr_to_all_nbr	1.36×10^{-3}
ratio_max_1_TFSF_nbr_to_all_nbr	2.72×10^{-4}

The least significant five features together only boost the accuracy by about 1%. It is also noted that not all the features are selected, as shown at the bottom of Table 3.4. These features are redundant as far as the diagnosis accuracy is concerned. In other words, given the existence of the top features, the addition of unselected features will not further boost the classification accuracy. In addition, it is observed that some features are quite effective according to both the Fisher score and SFS (*e.g.*, `resol_ratio`, and `ratio_failing_states_to_TFSF_ptn`); on the other hand, some features are ineffective (*e.g.*, `ratio_max_0_TFSF_nbr_to_all_nbr`, and `ratio_max_1_TFSF_nbr_to_all_nbr`).

Table 3.4: Sequential Forward Selection of the Features.

Selection	Feature name
1	<code>ptn_score</code>
2	<code>TFSP_ptn</code>
3	<code>bit_score</code>
4	<code>TFSP</code>
5	<code>resol_ratio</code>
6	<code>ratio_failing_states_to_TFSF_ptn</code>
7	<code>ratio_max_0_TPSF_physical_to_physical_nbr</code>
8	<code>ratio_passing_states_to_2^{nbr}</code>
9	<code>TFSF_ptn</code>
10	<code>ratio_failing_states_to_2^{nbr}</code>
11	<code>TPSF_ptn</code>
12	<code>ratio_max_0_TPSF_nbr_to_all_nbr</code>
13	<code>TPSF</code>
14	<code>passing_states</code>
15	<code>ratio_max_0_TFSF_physical_to_physical_nbr</code>
-	<code>failing_states</code>
-	<code>TFSF</code>
-	<code>nbr</code>
-	<code>ratio_max_1_TPSF_physical_to_physical_nbr</code>
-	<code>ratio_max_1_TPSF_nbr_to_all_nbr</code>
-	<code>ratio_max_1_TFSF_physical_to_physical_nbr</code>
-	<code>ratio_max_1_TFSF_nbr_to_all_nbr</code>
-	<code>ratio_max_0_TFSF_nbr_to_all_nbr</code>

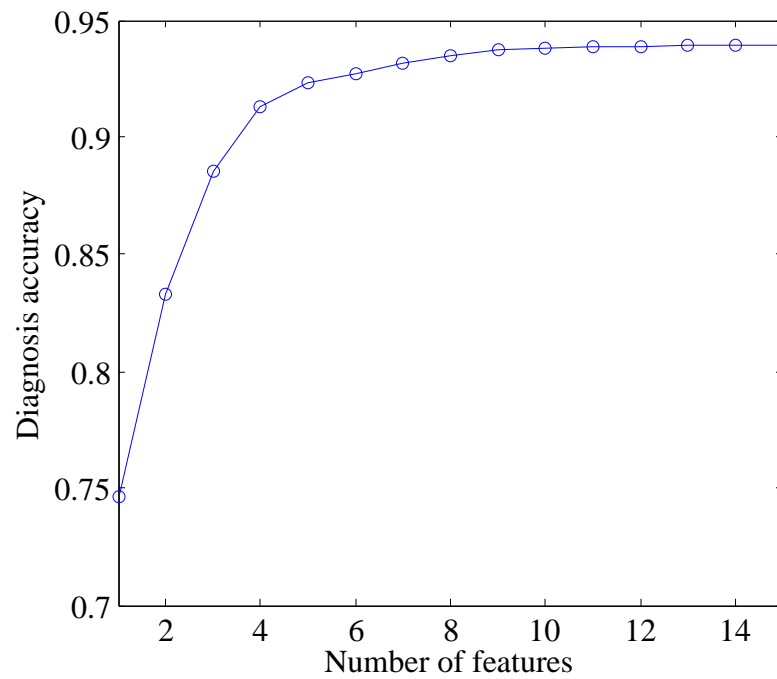


Figure 3.3: The diagnosis accuracy improves following the addition of each selected feature. The accuracy plot shows a sharp increase with the addition of the top few features. As more features are selected, the accuracy increase slows and eventually saturates before all the features are selected.

3.2.4 Resolution Improvement and Accuracy

A total of 14,710 candidates are produced from the initial diagnosis of all the virtual failing chips listed in Table 5.1. The number of incorrect candidates labeled by the first-level classifier is 8,005. Comparing these results with the injected locations reveals that all of these incorrect-labeled candidates are indeed incorrect.

A total of 6,705 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 procedure discussed in section II. Initially, the correct-labeled training set has 447 candidates and the incorrect-labeled training set has 3,197 candidates. After over-sampling the correct-labeled training set, both training sets have 3,197 candidates.

The PADRE prediction for all candidates are shown in Table 3.5. A total of 3,441 candidates are predicted as correct, of which 2,965 or 81.16% are indeed true candidates; for the remaining 11,269 candidates that are predicted as incorrect, 10,852 or 96.30% are indeed wrong candidates; overall, the candidate-level accuracy of PADRE reaches 93.93%. Note that among all 10,852 incorrect-labeled candidates, 8,005 are labeled by the first-level classifier, which has 100% accuracy. For the second-level classifier, it processes total 6,705 candidates, resulting in 2,965 correct-labeled candidates, and 2,847 incorrect-labeled candidates. Among them, 5,812 are correct, which results in an accuracy of 86.68% for the second-level classifier.

At the defect level, PADRE improves resolution for 3,276 or 96.86% of total 3,382 defects, and the remaining 106 defects have their original resolution retained. By comparing the improved resolution with the original resolution, it is clear that the resolution improvement is significant as shown in Figure 3.4a, where the red curve is the improved cumulative resolution and the blue curve is the original cumulative resolution. By comparing the improved resolution and the original resolution, the average defect-level resolution is shown to improve from 4.35 to 1.83. Moreover, the number of defects that exhibit ideal resolution increases from 447 to 2,798, an increase of more than $5\times$. In Figure 3.4b, each defect is plotted in a scatter plot according to its

improved resolution versus its original resolution. In the plot, all defects with ideal resolution would lie on the x-axis whereas defects without any resolution improvement would lie on the $y = x$ line. In addition, the accuracy of resolution improvement is also shown in the scatter plot. Defects with accurate resolution improvement are marked with blue circles, whereas defects with inaccurate resolution improvement are marked with red circles. It is shown that the resolution improvement for 3,249 or 99.2% of total 3,276 defects are accurate.

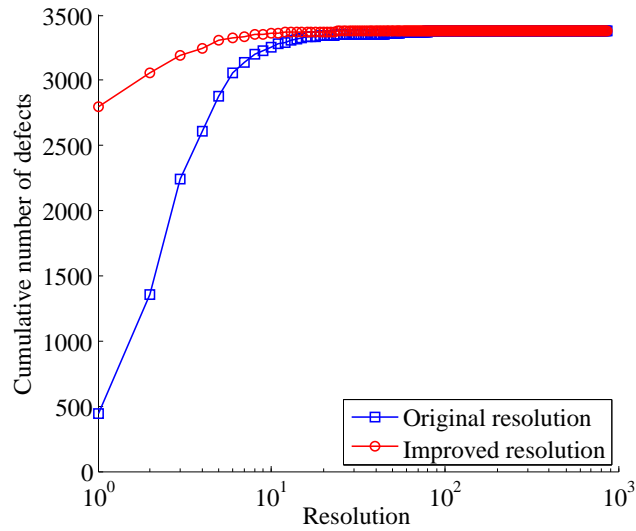
Table 3.5: PADRE Candidate-level Classification Statistics.

PADRE labeling		No. of candidates
Good	Correct	2,965
	Wrong	476
Bad	Correct	10,852
	Wrong	417

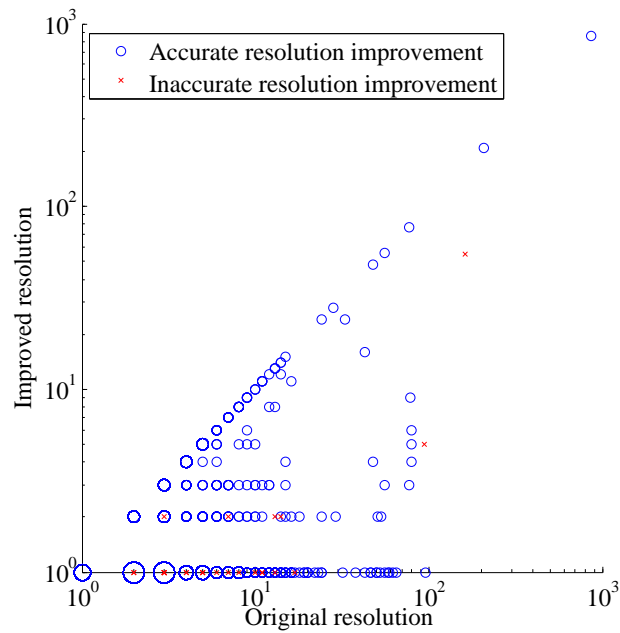
3.2.5 Dataset-Size Sensitivity

It is of our interest to know how PADRE performs in different scenarios, including the scenario when the failing data obtained in practice are limited in size. One major concern is that too few training data for the second-level classifier could hinder the performance and reliability of PADRE. To evaluate this issue, dataset-size sensitivity analysis is performed. Specifically, the experiment is performed by applying PADRE to some small sub-datasets, which are constructed from a small number of randomly-selected virtual failing chips.

In the experiment, the subsets are constructed in three different sizes, namely 25%, 50%, and 75% of all virtual failing chips. For each size, 10 different subsets are constructed through a random sampling process. PADRE is applied to all the subsets to evaluate the overall classification accuracy. The average classification accuracy for each size is shown in the *Within subset* column in Table 3.6. In Table 3.6, false positive represents the number of wrong candidates erroneously labeled as correct, whereas false negative represents the number of true candidates erroneously labeled as incorrect, and the total false is the sum of the two, i.e., the total number of mislabeled



(a)



(b)

Figure 3.4: (a) Resolution improvement through PADRE at the defect level shows that the number of defects exhibiting ideal resolution is increased by more than $5\times$. (b) Each defect is plotted according to its improved resolution versus its original resolution. Defects with accurate resolutions are marked with blue circles, whereas defects with inaccurate resolutions are marked with red crosses. The accuracy is maintained for 3,249 or 99.2% of all defects. Bubble size is proportional to the number of defects with the corresponding original/improved resolution.

candidates. Note that classification accuracy result for subset size of 100% is added to Table 3.6 as a reference. It can be seen from Table 3.6 that the even with only 25% size of original training sets, the overall accuracy drops by only about 5%, which is fairly robust.

In addition to examination of within-subset accuracy, the SVM classifier obtained from second-level classifier is applied to the entire dataset as well to obtain the overall PADRE accuracy on the entire dataset. The experiment on entire dataset allows us to see if the generality of the training set will be lost significantly for the case when only small training sets could be obtained. The PADRE accuracies of this setup are summarized under the *On entire dataset* category in Table 3.6.

It can be seen from Table 3.6 that the even with only 25% size of original training sets, the overall accuracy drops by only about 5% and 9% for within-subset accuracy and entire-dataset accuracy respectively. Though from the experiment, it is shown that PADRE is generally robust against data size change in both cases when there is only limited initial data or when there is a large data size but only a small number of training data could be obtained. However, it should be noted that the drop in accuracy is not smooth as shown in Table 3.6, and even at 25% original size, the subset still holds more than 400 chips. It is also clear from Table 3.6 that the entire-dataset accuracy suffers more than the within-subset accuracy as the training sets shrink, which agrees with our expectation, as with the size of training sets shrinking, it is expected that randomness would play a higher role in the generality of the training sets, depending on the distribution of the candidates. From the results, it appears that having a small initial dataset may not be as bad as failing to extract enough training data.

3.3 Experiments with Silicon Data

PADRE is also applied to actual failing chips to evaluate its performance. Two different silicon data sets are investigated. The first data set consists of actual failing chips of the same industrial design that is used for the virtual data experiment. The second data set consists of a large number

Table 3.6: Dataset Sensitivity Analysis with Reduced Subsets for Training.

Subset size	Total chips	Total candidates	Within subset				On entire dataset			
			false positive	false negative	total false	accuracy	false positive	false negative	total false	accuracy
100%	1,606	14,701	476	417	893	93.93%	476	417	893	93.93%
75%	1,204	11,256	363	333	696	93.82%	538	520	1,058	92.80%
50%	803	7,793	69	886	955	87.75%	1,577	605	2,182	85.16%
25%	401	4,300	70	410	480	88.84%	1,566	611	2,177	85.19%

of failing chips from a second industrial design. Both data sets have been tested and diagnosed using commercial tools to produce candidates. PADRE is applied to the candidates to improve the diagnostic resolution following the same procedure as in the virtual data experiment.

Verifying both the resolution improvement and accuracy for virtual data is straightforward since the injected faults are known. Unfortunately, this is not the case for most of the silicon data sets. However, we do have in hand five chips from the second data set that have been PFAed, and we are able to verify the resolution improvement accuracy for these chips.

Data Set One

The first silicon data set consists of 360 failing chips and 507 defects, with a total of 1,936 candidates. This is a relatively small data set compared to the virtual data set, which has more than 14,000 candidates.

Due to the small number of candidates per defect in this data set, we have to adjust the candidate number threshold used for training set construction. Specifically, the number of candidates per defect for the incorrect-labeled training set Q is reduced from 20 to 10 in order to collect sufficient training data. After the threshold adjustment, the initial correct-labeled training set has 161 candidates, and the initial incorrect-labeled training set consists of 501 candidates.

The first-level classifier predicts 764 candidates as incorrect, and the second-level classifier predicts 402 candidates as correct. The resolution improvement is shown in Figure 3.5. The resolution at the defect level is improved by 24.6% from 3.82 to 2.88. The numbers of defects that exhibit ideal resolution is increased by 77.6% from 161 to 286.

Data Set Two

The second silicon data set consists of 5,362 failing circuits, each of which contains 2,309 logic gates. Each circuit has been tested with 5,362 production tests. After initial diagnosis, a total of 17,786 defects and 36,186 candidates are reported, which is approximately twice the size of the

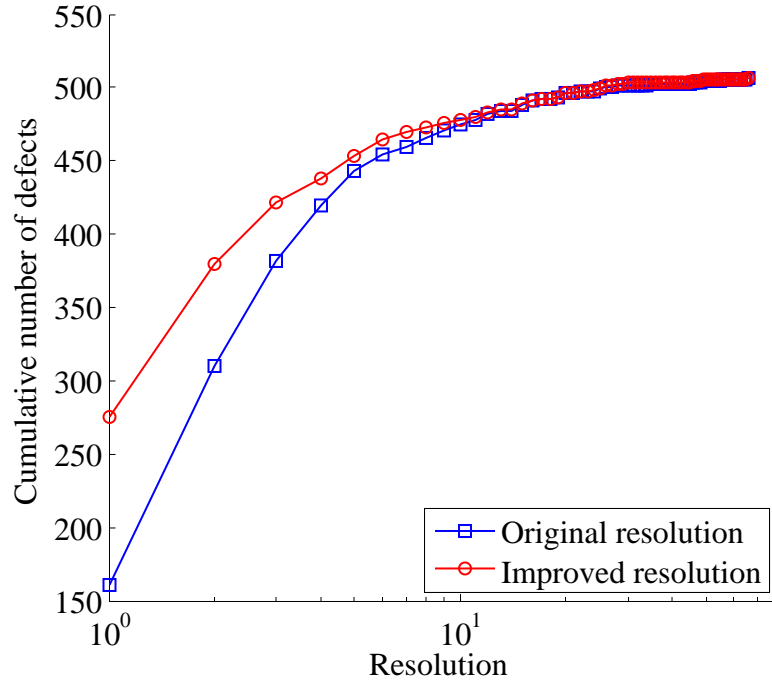


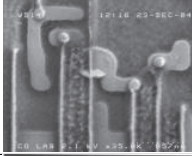
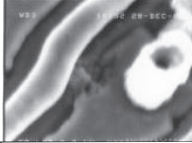
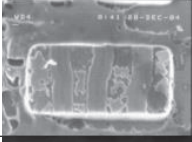
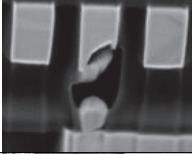
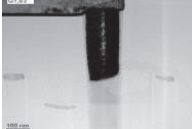
Figure 3.5: Resolution improvement for the first silicon data set shows the number of defects that exhibit ideal resolution is increased by 77.6%.

virtual data set.

For the second data set, the incorrect-labeled training set is constructed with $Q = 20$ as in the virtual data set experiment. An initial correct-labeled training set with 5,519 candidates and an initial incorrect-labeled training set with 7,376 candidates are constructed. In the resolution improvement process, the first-level classifier removes 16,836 incorrect-labeled candidates, and the remaining 19,350 candidates are processed by the second level. The resolution improvement result can be seen in Figure 3.6, where the numbers of defects exhibiting ideal resolution is increased by more than 4,000.

For this data set, we have five chips that have been PFAed, the original resolution and improved resolution of these five chips are shown in Table 3.7. Overall, the resolution of three chips are improved, by comparing the true candidate as revealed by PFA and the prediction made by PADRE, it is also shown that resolution improvement of all the chips are accurate.

Table 3.7: Resolution Improvement of PFAed Failing Chips.

SEM of failure	Original resolution	Improved resolution	Accurate?
	16	4	Yes
	1	1	Yes
	4	3	Yes
	11	3	Yes
	2	2	Yes

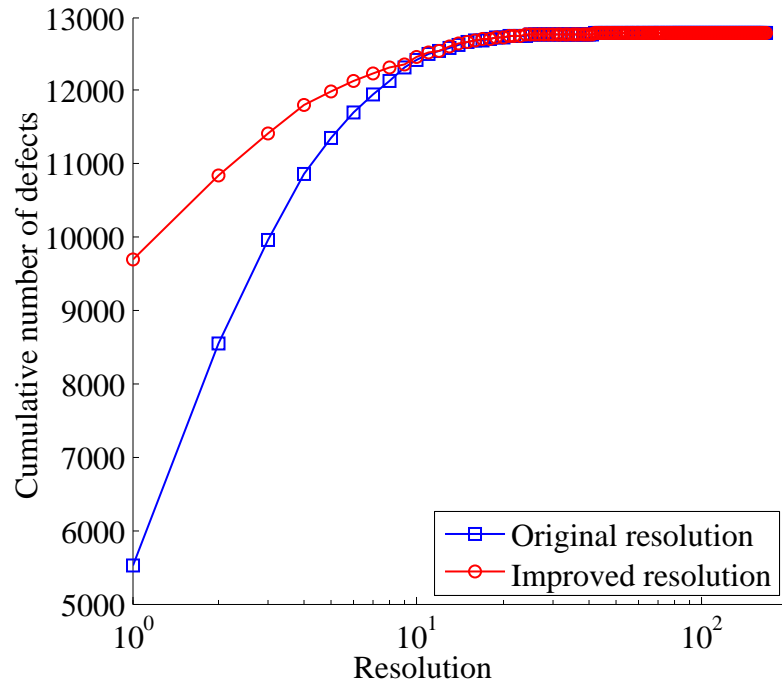


Figure 3.6: Resolution improvement for the second silicon data set shows that the number of defects exhibiting ideal resolution is increased by nearly $2\times$.

3.4 Computational Cost

PADRE does not require any additional testing or design modification, it can be easily applied as an add-on resolution improvement technique to any diagnosis approach with little additional cost. The cost associated with PADRE is mostly computational cost, and the computational time overhead associated with PADRE can be divided into three main parts as follows:

- Feature extraction
- Classifier construction
- Classification

The most time-consuming part among the three is feature extraction. Specifically, it usually takes significantly more time to extract physical features as compared to the logical features. Extracting logical features requires only the analysis of failing logs, diagnosis callouts and fault simulation of the candidates, whereas extracting physical features requires additional analysis of

the netlist and circuit layout.

The classifier construction and classification times are almost negligible. For our virtual data set with more than 14,000 candidates, the classifier is constructed in less than 9 seconds running on a workstation with 8 GB memory and a 2.60 GHz duo-core processor. The classification is completed in less than 5 seconds. It should be noted that the classifier construction time grows at the order of $O(n^3)$, where n is the number of training candidates[67], thus it may take longer time to construct classifier for a significantly larger data set.

3.5 Summary

In this chapter, a machine learning-based resolution improvement technique, PADRE is presented. Unlike other machine learning-based techniques, PADRE constructs correct-labeled and incorrect-labeled training sets from unlabeled candidates, instead of relying on historical data or simulation data. PADRE is evaluated with comprehensive experiments. In the experiments, PADRE is shown to improve resolution significantly for both the simulation dataset and the silicon dataset. Specifically, the number of defects with perfect resolution is increased by $5\times$ for simulation experiment, and $2\times$ for silicon experiment.

Chapter 4

AL PADRE

4.1 AL PADRE Methodology

Active learning (AL) helps to mitigate the conflict between limited PFA resources and the large amount of failed defects by querying the most informative defects for improving the classification accuracy of PADRE. In each AL iteration, from a population of diagnosed defects that are qualified for PFA by other necessary criteria (*e.g.*, defects that have no more than five candidates), the most valuable defect is identified by one of two selection methods; namely *discrepancy check* and *within-margin*. Then the labels for the candidates of the queried defect are obtained through a successful PFA, which are in turn fed back to PADRE as training data for updating the classifier. This iterative process continues until PFA resources are exhausted, or when no more defects satisfy the selection criteria. It should be noted that although PFA is always assumed to be successful by AL PADRE, in practice, a PFA can be unsuccessful. AL PADRE copes with an unsuccessful PFA by simply not updating the training data. It should also be noted that AL PADRE can benefit from defects selected for PFA for conventional reasons (*e.g.*, uncovering root-cause for a large portion of a failure Pareto). For such cases, the resulting labeled candidates

would simply be added to the existing training data for re-learning the classifier¹.

It is clear that the effectiveness of AL PADRE will significantly depend on the method used for selecting a defect with the most informative candidates for improving the classifier. Conventionally, AL can be categorized into either pool-based sampling [68, 69] or stream-based selective sampling [70]. In stream-based selective sampling, it is assumed that the unlabeled instances are made available one by one, which implies that AL has to determine for each instance whether to query the label of the instance. On the other hand, in pool-based sampling, it is assumed that unlabeled instances are all simultaneously available. In this situation, AL must choose what is considered the most informative from the pool of unlabeled instances. In pool-based AL, one effective selection criterion could be a pre-constructed uncertainty region where the existing classification is still ambiguous, so that the label of any instance within the region could adjust the existing decision boundary. Other query criteria adopt pre-defined informative metrics, so that instead of a pure random selection, a biased selection according to such informative metrics is made [71]. For diagnosis and PFA, subsets of instances are available for labeling which means that neither pool- or stream-based sampling is directly applicable. For the two selection methods used by AL PADRE, within-margin is a modified pool-based selection method and discrepancy check is a novel selection method developed specifically for diagnosis and PFA. The two selection methods target two different types of defects that can potentially improve the existing classifier. Discrepancy check targets a defect with a large number of correct-labeled candidates; within-margin targets a defect that has a large number of candidates that reside within the decision margin of the classifier. The goal of these two approaches is the same, that is, to use PFA to identify labeled instances that minimizes the likelihood of misclassification by PADRE. The two types of defects targeted are both likely to contain misclassified candidates, as explained later, by re-training PADRE with the correct labels of these candidates, other similar candidates are expected to be correctly classified.

¹Re-learning the classifier when the training data is modified/updated is a trivial matter, taking just a few seconds using a typical compute server.

4.1.1 Discrepancy Check

The discrepancy check intends to identify a defect with the largest number of potentially mislabeled candidates. The discrepancy between the number of expected correct candidates and the number of labeled- or predicted-correct candidates provides the lower bound on the number mislabeled candidates for a given defect. This is based on the assumption that there is only one correct candidate corresponding to each defect. Therefore, any discrepancy must arise from either labeling multiple candidates as correct for the defect, or failing to label a single candidate as correct. To illustrate the discrepancy check, consider the example shown in Table 4.1. In this example, a chip with three defects is diagnosed, and there are three candidates reported for each defect. For each candidate, the label predicted by PADRE and its true label are listed in column 3 and 4, respectively, with “1” representing a correct-labeled candidate and “0” representing an incorrect-labeled candidate. If the PADRE label is the same as the true label, the candidate is said to be labeled accurately. For each defect, the discrepancy is therefore the difference between the total number of labeled-correct candidates and the expected number of correct candidates, which of course is just one. The discrepancy is calculated as:

$$\text{Discrepancy} = \text{No. labeled-correct candidates} - 1 \quad (4.1)$$

Accordingly, defect 1, 2, and 3 each has a discrepancy of 1, 1, and 0, respectively. However, as shown by the candidate accuracy, the number of inaccurate candidates for the three defects are 1, 3, and 2, respectively. Defects 2 and 3 both have more inaccurate candidates than their discrepancy indicates. This is because the discrepancy only provides a lower bound on the possible number of inaccurate candidates, since it is impossible to know the exact number of inaccurate candidates without knowing the actual candidate labels. Still, a defect with a large discrepancy is certain to contain an also large number of inaccurate candidates. By revealing the true labels of the inaccurate candidates through PFA, the AL algorithm is able to effectively improve the accuracy of the PADRE classifier.

Table 4.1: Discrepancy Check.

Defect	Candidate	PADRE label	True label	Candidate accurate?	Defect discrepancy
1	1	1	1	Yes	1
	2	1	0	No	
	3	0	0	Yes	
2	1	0	1	No	1
	2	1	0	No	
	3	1	0	No	
3	1	0	1	No	0
	2	0	0	Yes	
	3	1	0	No	

Discrepancy is a direct indication that there are wrongly-classified candidates. Specifically, there could be two types of misclassifications, namely, it could be that the actual true candidate is misclassified as incorrect, or alternatively, it could be that some incorrect candidates are misclassified as correct. A successful PFA corrects both types of misclassification.

4.1.2 Within-Margin

PADRE uses SVM, a binary classifier that labels the candidates as either correct or incorrect. As expected, the candidates are not always correctly labeled, in addition, some of the candidates are more likely to be misclassified than others. The decision boundary that determines classification is defined by the support vectors, which are correct and incorrect training candidates with similar features. The support vectors are selected to maximize the boundary margin between correct- and incorrect-labeled training candidates.

The candidates within the margin are more likely to be wrongly classified than candidates outside the margin. There are an infinite number of decision boundaries that can separate the correct and incorrect training candidates, but it is not known which boundary is optimal for the entire data set. The initial boundary may be far from optimal. For example, Figure 4.1 illustrates two data sets separated by a non-optimal decision boundary learned from a subset of labeled data. The candidates outside the margin belong to the well-defined two classes, whereas the candidates

within the margin are inherently more ambiguous since they are close to both classes. As illustrated in Figure 4.1, if the unlabeled candidates within the margin can be labeled, the decision boundary can be adjusted to become optimal. On the other hand, if only the unlabeled candidates outside the margin are labeled, the existing boundary will only change slightly, because most of them are already correctly classified.

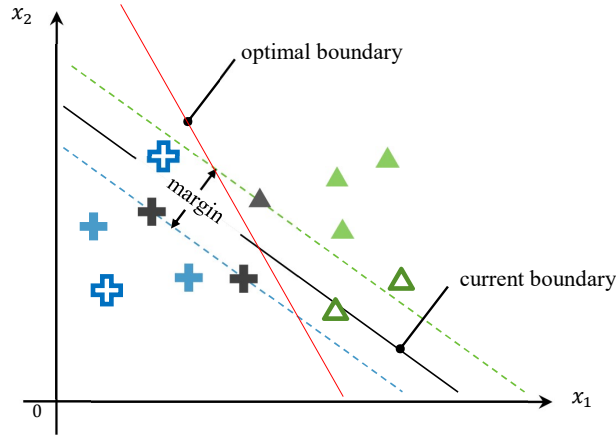


Figure 4.1: Two data sets (crosses and triangles) are plotted with respect to two features, x_1 and x_2 . Solid crosses and triangles are labeled training data from two classes, while unfilled crosses and triangles are unlabeled data. The support vectors (grey crosses and triangles) define the current boundary and its margin. Labeling the unlabeled data within the margin leads to a new, optimal boundary.

For a linear SVM, a candidate d_i is defined as being within the boundary margin if

$$\|\mathbf{w}^\top \mathbf{x}_i + c\| < 1 \quad (4.2)$$

where \mathbf{x}_i is the feature vector of d_i , \mathbf{w} is the weight vector for each feature, c is a constant that is learned by SVM, and $\|\bullet\|$ denotes the ℓ^2 -norm of a vector. These candidates are closer to the boundary than the support vectors, and therefore are considered ambiguous in terms of their classification result. The closer a candidate is to the decision boundary the more likely that it is misclassified. A candidate located exactly on the decision boundary, *i.e.*, $\mathbf{w}^\top \mathbf{x}_i + c = 0$, has a 50% likelihood of being misclassified. Therefore, a defect that has a large number of within-margin candidates is a defect that could potentially have a large number of misclassified

candidates.

4.1.3 Stopping Criteria

AL is an iterative process of repeatedly requesting new training data. There is a point, however, when no additional training data can further improve the classification accuracy. Certainly the ultimate stopping criteria is when all data is labeled or when PFA resources are exhausted. If the AL criteria are correctly formulated, there should be a diminishing return with additional training data however. As the classifier converges to the optimal classifier, a smaller and smaller change in the classifier is expected from consecutive iterations of AL. To track the converging classifier, the weight vector from the SVM classifier is extracted. Again, the weight vector, w , is the weight assigned to each candidate feature. So the change in weight vector directly reflects a change in the classifier. Weight vectors w_1 and w_2 between two consecutive iterations of AL are compared using *cosine similarity*, which is defined as follows:

$$Similarity = \frac{w_1^T w_2}{\|w_1\| \|w_2\|} \quad (4.3)$$

where $\|\bullet\|$ denotes the ℓ^2 -norm of a vector.

Cosine similarity varies between -1 and 1. When w_1 and w_2 are exactly the same, the cosine similarity is 1; and when they are completely opposite, the cosine similarity is -1. In this work, a similarity value of 0.9999 is used as the stopping criterion for AL.

Another stopping criteria related to PFA is also adopted. If the potential benefit of PFA is low, then PFA can be avoided so as to save cost. The potential benefit of PFA for a given defect is reflected by its discrepancy count, or its within-margin-candidate count. When the counts are low, it is an indication that even a successful PFA will lead to a low number of corrected misclassifications, leading to little improvement in classification accuracy. The threshold of what is considered low depends on the availability of resources for PFA. If the resources are abundant, one may even choose to continue PFA as long as the discrepancy count is greater than zero.

When both the aforementioned stopping criterion are satisfied, or when the PFA resources are exhausted then AL PADRE is completely halted. A low count (*i.e.*, a value of two) for discrepancy check is used alone however when switching from using discrepancy check to within-margin. The reason the cosine similarity is not used to trigger a transition is that for a few rare cases, it has been observed that cosine similarity triggers a switch that is too early. This occurs because early iterations in AL encounter a small subset of new training data that does not significantly alter cosine similarity. For such cases, the algorithm should not transition to within-margin, because subsequent use of discrepancy check does lead to substantially changing of the classifier.

4.1.4 Implementation

To ensure optimal and consistent performance within AL PADRE, there are a few implementation issues to be carefully addressed.

Defect Tiebreaker

For both the discrepancy check and within-margin, it is possible that multiple defects have the same count for discrepancy or within-margin. In case of a tie, the defect with the largest number of candidates is selected. Breaking a tie in this way is justified since a successful PFA will provide all the labels for all the candidates associated with the selected defect. Thus, a defect with more candidates can provide more correct-labeled or incorrect-labeled data for updating the classifier. For cases where multiple defects are tied for both the discrepancy and the number of candidates, a random selection is made.

Training Data Update

As discussed in Section II, the training data are balanced in PADRE, *i.e.*, the correct-labeled candidates are oversampled to match the size of the incorrect-labeled candidates with a random

duplication process. To maintain balance of the training data, additional training data obtained from PFA must be balanced as well. One straightforward approach is to combine the new correct- and incorrect-labeled candidates with the existing data, and then perform oversampling to balance the augmented data set. However, this approach leads to rapid fluctuation in accuracy due to the resulting randomness in the training data. Worse, due to the randomness in the oversampling process, some of the new training data may not be well represented in the balanced sets, which undermines the value of the new data obtained from PFA.

To address this issue, for each iteration of AL, the new correct- and incorrect-labeled candidates are first balanced among themselves by randomly duplicating the correct-labeled candidates to match the number of incorrect-labeled candidates. Then the new, balanced training data are added to the existing training sets. This approach not only ensures balanced data sets, but also that the composition of the training data are consistent from one iteration of AL to the next, since the exact training data used by one iteration is carried over to the next.

PFA Candidate Weight

Typically only a small number of candidates are labeled by a single successful PFA, due to the fact that only a defect with less than five candidates is typically selected for PFA. In contrast, there is typically a large number of candidates in the initial training set. If the new data resulting from PFA are simply added to the existing training set, it will be overwhelmed by the larger existing data, leading to little impact on a re-learned classifier. To address this issue, new candidates resulting from PFA are weighted higher than existing candidates, *i.e.*, each new candidate is duplicated multiple times. The challenge, however, is to find the right weight. If the weight is too large, the classifier will overfit the new PFA candidates. On the other hand, if the weight is too small, the classifier will not change due to the new candidates, resulting in a prolonged process for improving classification accuracy, possibly exceeding the PFA resources too quickly. The weight is a variable that depends heavily on the specific data set involved. For the data sets

used in our experiment, by trial and error, we found that a weight between 10 and 50 demonstrates accuracy improvement without incurring overfitting. As a result, all PFA candidates are weighted $20\times$ more than other candidates.

With all these implementation issues addressed, the overall flow of AL PADRE is given in Algorithm 1. It should be emphasized that while AL PADRE features an automatic stopping mechanism, it does not imply that PFA, an indispensable step in yield learning, should stop. For AL PADRE, a stop implies that PADRE has reached a stable, high-level accuracy for the current data set, and any additional PFA will not significantly further improve the accuracy. Therefore, after AL PADRE indicates a stop, PFA should still be carried out for the purpose of other yield-learning analyses. When there is a change in the diagnosis data set, *e.g.*, when new diagnosis data become available, AL PADRE should be executed again to update the classifier to the latest data set.

4.2 Simulation Experiments

AL PADRE is validated with comprehensive simulation experiments. The data used in the simulation experiments mimic real failed chips that are affected by various types of defects. The chips with injected defects are virtually tested and diagnosed using a commercial flow. AL PADRE is applied to refine the diagnostic resolution and to identify defects for PFA. Accuracy is tracked with each PFA to evaluate the effectiveness of AL PADRE. Specifically, the accuracy is evaluated against *all* the candidates, by contrasting the labels predicted by PADRE and the true labels. It should be noted that candidates used for training the SVM classifier are not excluded from accuracy calculation, because the true labels of the candidates used for training are in fact unknown to PADRE, which only assumes the labels based on heuristics.

Algorithm 1 AL PADRE.

```
while remaining PFA resources do
  Execute PADRE
  Identify PFA-qualified defect (i.e., #candidates  $\leq 5$ )
  Calculate the max discrepancy in qualified defects
  if max discrepancy  $> 2$  then
    Method = discrepancy check
  else
    Calculate the max within-margin in qualified defects
    if max within-margin  $> 2$  then
      Method = within-margin
    else
      break
    end if
  end if
  Select PFA defect with Method
  Obtain true labels of defect candidates
  Balance new training data
  Update PADRE training data
  Re-learn classifier
  Calculate Similarity
  if Method==within-margin and Similarity $>0.9999$  then
    break
  end if
end while
```

4.2.1 Setup

An industrial design is used as the chip under test. The chip used in this experiment is an ASIC manufactured in 130 nm technology that contains about one million gates. The production test set includes 36 scan-chain tests followed by 3,403 logic tests that together achieve 99.5% SSL fault coverage.

More than 1,600 instances of multiple types of faults are randomly injected into the circuit to emulate defective chips. The faults used to emulate the defects are simulated using a commercial tool. The injected faults include: and-, or-, and four-way bridge faults, input pattern faults[66], SSL faults, and MSL faults. The number of injected faults of each type are listed in Table 5.1.

Table 4.2: Faults Types Injected into the ASIC Design.

Fault type	Number of chips
AND bridge	96
OR bridge	94
Four-way bridge	275
Input pattern	412
SSL	364
MSL	365

The fault types listed in Table 5.1 are chosen to represent the large variety of actual defects that occur in real chips. The advantage of using a virtual population of failed chips is that it provides a large amount of labeled data, *i.e.*, failed chips with known defect locations.

Each defective chip is tested and diagnosed. Because each chip is virtual in nature, *i.e.*, we know the location of all defects, the accuracy of the diagnosis resolution improvement is easy to verify. However, a list of “golden answers” is never available in practice.

Two sets of experiments are designed to evaluate AL PADRE under different scenarios². The first set of experiments evaluate AL PADRE assuming a large number of failed chips but only a small number of initial training data. In this case, we use only 25% of the available training data to train an initial classifier. Then AL PADRE is applied to improve the accuracy of the initial

²These two scenarios are based on extensive discussion with industrial collaborators that have vast experience with performing PFA on in-production ICs.

classifier, and its performance is evaluated against a baseline performance that uses a practical, random selection approach. To mimic a real-world scenario, only defects with no more than five candidates are selected for PFA. However, in one experiment setup, this requirement is relaxed to evaluate the full potential of AL PADRE.

The second set of experiments evaluate AL PADRE assuming a production environment, where every week a number of diagnosed chips stream in, and a subset of them can be selected for PFA. Note that due to the destructive nature of PFA, for a failed chip with multiple defects, it is assumed only one of the defects can be PFA'ed. Each week, AL PADRE, based on both the data newly collected and the data from previous weeks, has to decide which defects should be selected for PFA. In the experiments, the number of diagnosed chips per week is assumed to be 40, and the PFA is restricted to five per week. The AL PADRE performance is evaluated against two baselines, the first is against performing PFA using random selection, and the second is against performing no PFA at all. In addition, to examine different starting scenarios, in one setup of the experiment, the first four weeks of diagnosis data are accumulated to learn the initial PADRE classifier; whereas in another setup, this arrangement is relaxed, so that the AL PADRE is applied from week one.

4.2.2 Results

For the first set of experiments, the initial classifier is trained with an initial training set of 111 correct-labeled candidates and 799 incorrect-labeled candidates, and achieves an accuracy of 77.0%. AL PADRE is subsequently applied for PFA defect selection from all the defects that satisfy the candidate limit criterion. The result obtained from AL PADRE is shown in Figure 4.2; it is observed that after some fluctuation from the first few PFAs, the accuracy increases from an initial 77.0% to a stable 90% level after 25 PFAs. While remaining accurate, the AL PADRE transits from discrepancy check to within-margin after 37 PFAs, and after another 17 PFAs using within-margin, AL PADRE terminates with an accuracy of 90.2%.

In addition to the accuracy improvement, AL PADRE also improves the overall resolution. The resolution distribution is shown in Figure 4.3. Originally, the diagnosis data have 447 defects with ideal resolution of one, PADRE significantly increases this number to 2,584, which is more than $5\times$. AL PADRE further improves the number of defects with ideal resolution by 107 to 2,691.

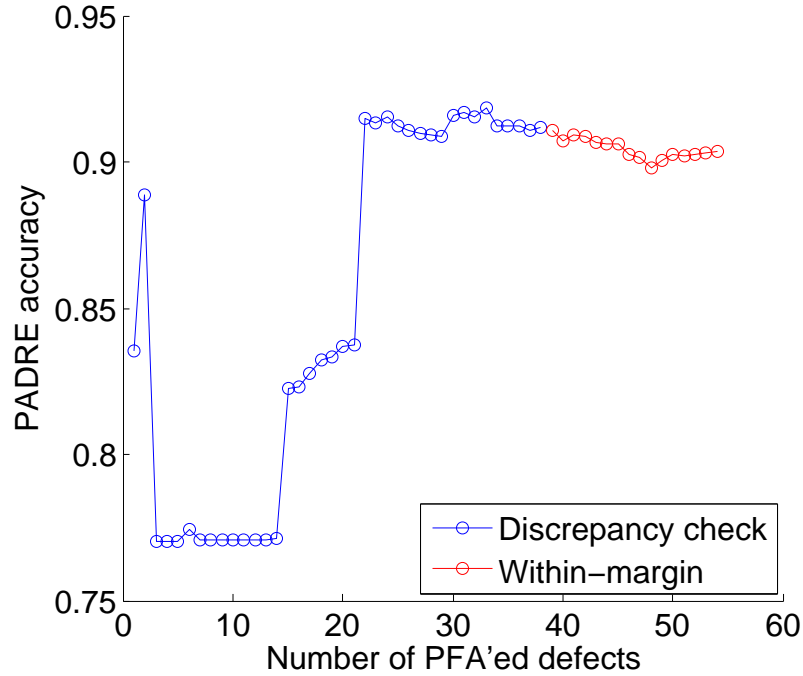


Figure 4.2: PADRE accuracy versus the number of PFA'ed defects selected by AL PADRE. The initial accuracy is 77.0%.

The performance of AL PADRE is compared against a baseline approach that uses random selection, where the only difference is that instead of using AL PADRE, each PFA defect is randomly selected from all the defects that satisfy the candidate-limit criterion. The baseline experiment comprises 100 PFAs in order for accuracy to increase to a comparable level with AL PADRE. Considering the intrinsic variation associated with random selection approach, the experiment is repeated 100 times to produce an average result. PADRE accuracy over 100 experiments are plotted in Figure 4.4a, with the average plotted as the bold blue curve. It can be seen that, with random selection, the accuracy increases to above 90% after more than 70 PFAs,

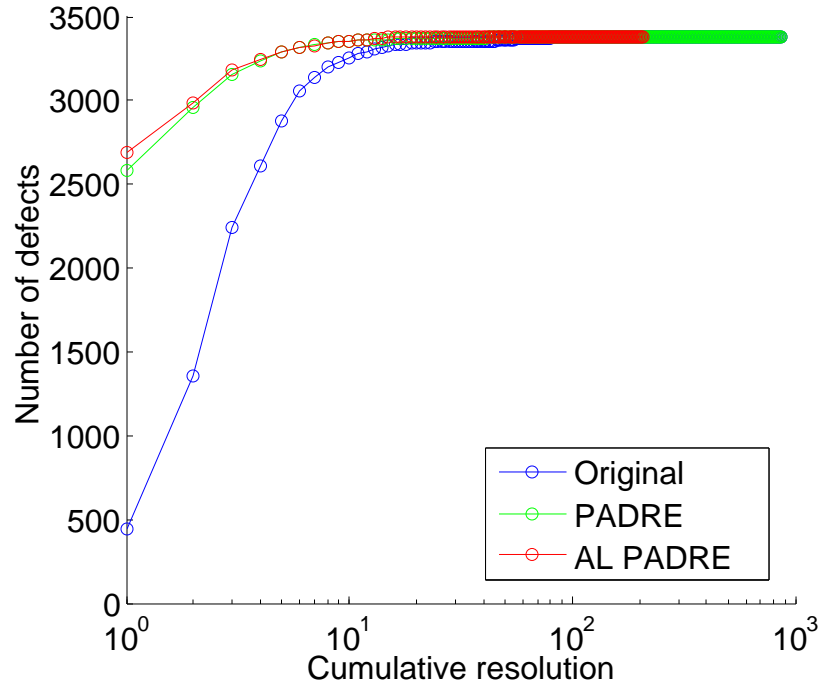


Figure 4.3: The cumulative diagnostic resolution distribution shows that PADRE improves the number of defects with ideal resolution by more than $5\times$ over the original resolution, and AL PADRE further improves the number by 107. Each point on the plot shows the number of defects that have a total number of candidates no greater than the given cumulative resolution.

on average. In contrast, as shown in Figure 4.4b, with AL PADRE, only 25 PFAs are needed to reach 90% accuracy level.

The full potential of AL PADRE is evaluated in an experiment of similar setup only without the candidate-limit restriction, meaning that any defect can be selected for PFA. The accuracy plot is shown in Figure 4.5; it is observed that accuracy saturates at around 90% after just two defects are PFA'ed, and the total number of PFA'ed defects is only eight, notably all are selected using discrepancy check.

The second set of experiments assume a production environment, with four weeks of accumulation. The AL PADRE accuracy plot is shown in Figure 4.7. It can be seen that in comparison with the two baseline approach performances, PFA using AL PADRE exhibits consistently higher accuracy starting from the first week of PFA. On average, the AL PADRE accuracy is

2.2% higher than PFA with random selection, and 3.6% higher than no PFA at all. Note that as the result of new incoming data each week, the accuracy may not keep increasing monotonically over the weeks, as existing classifier may not always accurately classify the new data. However, PFA with AL PADRE helps mitigate the fluctuation, because AL PADRE can focus PFA resources on investigating the new defect candidates that do not comply well with the existing classifier. For the setup of no accumulation week, the similar result of the three processes is shown in Figure 4.7. On average, AL PADRE accuracy is 2.1% higher than selecting chips for PFA randomly, and 3.1% higher than performing no PFA at all.

4.3 Silicon Experiment

AL PADRE is also used in a silicon experiment, that is, it is applied to actual failed chips to evaluate its performance. The silicon data stems from a commercial chip fabricated using state-of-the-art technology, and the failed chips are tested and diagnosed using a commercial flow. The data set consists of 353 defects and 2,462 candidates. Among the 353 defects, 19 are PFA'ed. For the silicon data, unlike simulation-based data set, the actual labels of most of the candidates are unknown, except for the few (*i.e.*, 19) that are revealed by PFA.

In order to evaluate the performance of selecting chips for PFA using AL PADRE against the actual chip-selection decisions made by a typical manufacturer, an oracle that is aware of the true labels of all the candidates is required. For all the candidates, except for those from the 19 PFA'ed defects, labels are predicted by a PADRE model which is trained using all available training data. These predicted labels form the oracle.

Then similar to the first simulation-based experiment, 25% of the available training data is used again to construct an initial classifier. By comparing the labels predicted by the initial classifier and the oracle, the accuracy of the initial classifier is found to be 61.2%. AL PADRE is then applied to the initial classifier, and again only defects with no more than five candidates are selected for PFA. The result is shown in Figure 4.8, it is observed that using AL PADRE, the

accuracy increases from an initial 61.2% to above 85.0% after only two PFA'ed defects, and the accuracy reaches 87.2% after 19 PFA'ed defects. On the other hand, updating the initial classifier using the 19 actual PFA'ed defects selected by the manufacturer, the accuracy is increased to above 85.0% after 13 PFA'ed defects, and the accuracy reaches 85.3% after 19 PFA'ed defects. Thus AL PADRE virtually matches this accuracy with more than $6\times$ fewer PFA'ed defects.

In addition to better accuracy improvement, using AL PADRE is also found to better improve the resolution than using the actual PFA'ed defects. As can be seen in Figure 4.9, using the actual PFA'ed defects improves the number of defects with ideal resolution from 88 to 107, whereas using AL PADRE, the number is increased from 88 to 123, which is 84% more than what is achieved with the actual PFA'ed defects.

4.4 Summary

In this chapter, AL PADRE is presented. AL PADRE extends the capability of PADRE with a novel, active learning-based PFA selection approach. AL PADRE selects the most informative defects for PFA for improving diagnostic resolution. AL PADRE is validated by both simulation experiment and silicon experiment. Simulation experiments show that by using AL PADRE, the number of physical analyses required for increasing the accuracy to 90% is reduced by more than 60% on average compared to baseline approach. In the silicon experiment, by using AL PADRE, the number of chips needed to undergo PFA was reduced by more than $6\times$ in order to increase diagnosis accuracy by more than 20%.

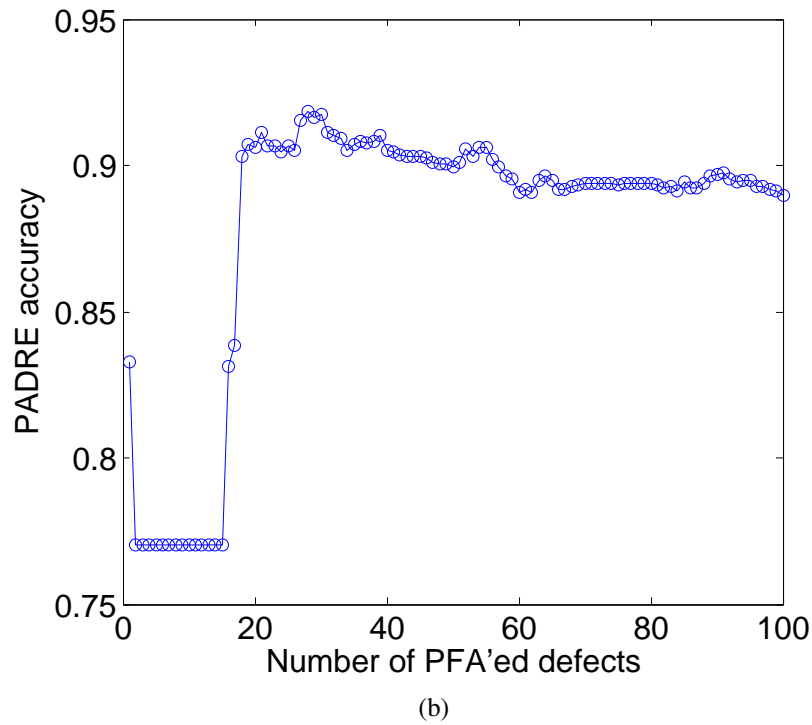
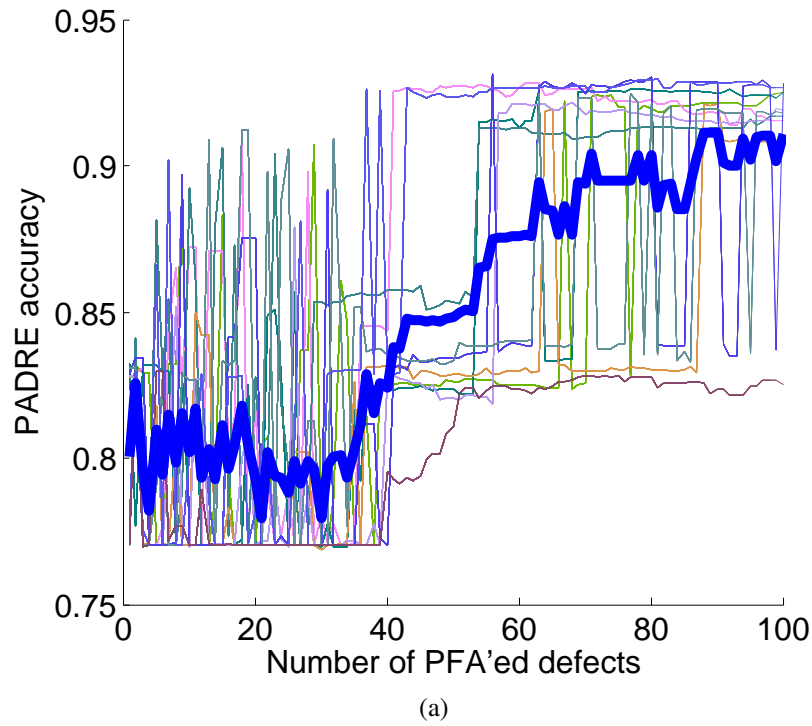


Figure 4.4: PADRE accuracy versus the number of PFA'ed defects selected by (a) random selection and (b) AL PADRE. For random selection, the average accuracy of the 100 experiments is plotted as a bold blue curve. The initial accuracy is 77%.

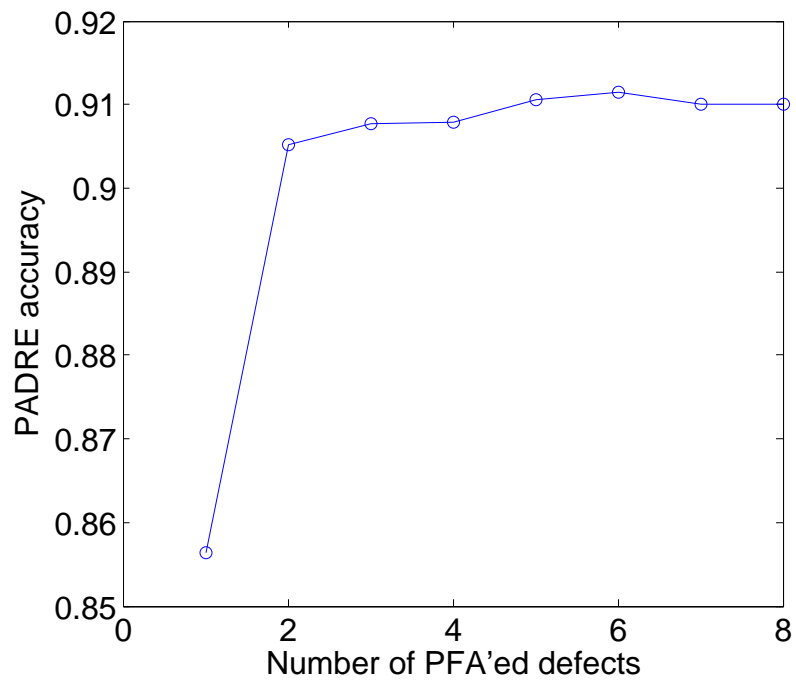


Figure 4.5: PADRE accuracy versus the number of PFA'ed defects selected by AL PADRE when defects are selected without the five-candidate restriction.

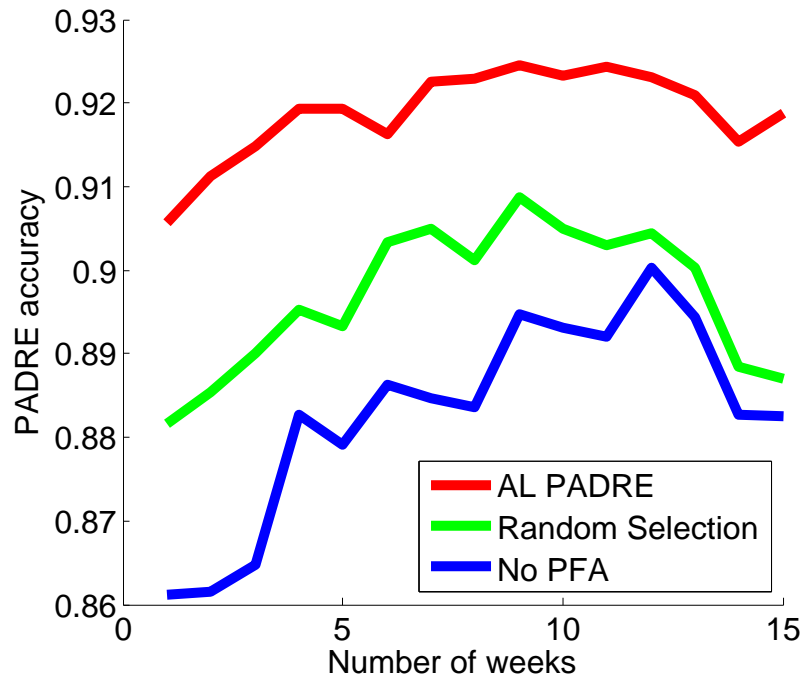


Figure 4.6: PADRE accuracy versus the number of weeks of PFA using three different processes, namely, PFA with AL PADRE, PFA with random selection and no PFA. Note that the first four weeks of accumulation is not shown in the plot, as no PFA is performed during that period.

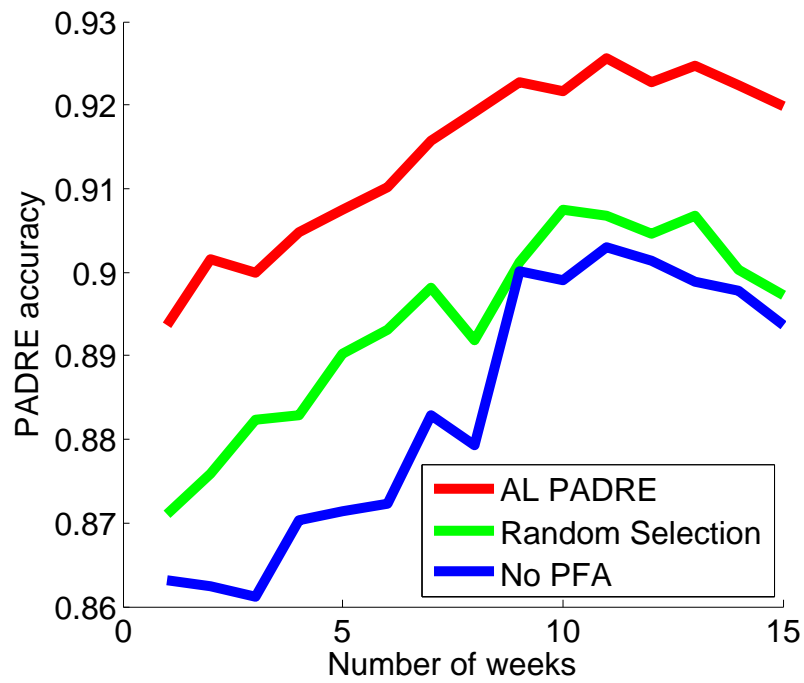


Figure 4.7: PADRE accuracy versus the number of weeks of PFA using three different processes, namely, PFA with AL PADRE, PFA with random selection and no PFA. Note that there is no accumulation in this setup, all three start from week one.

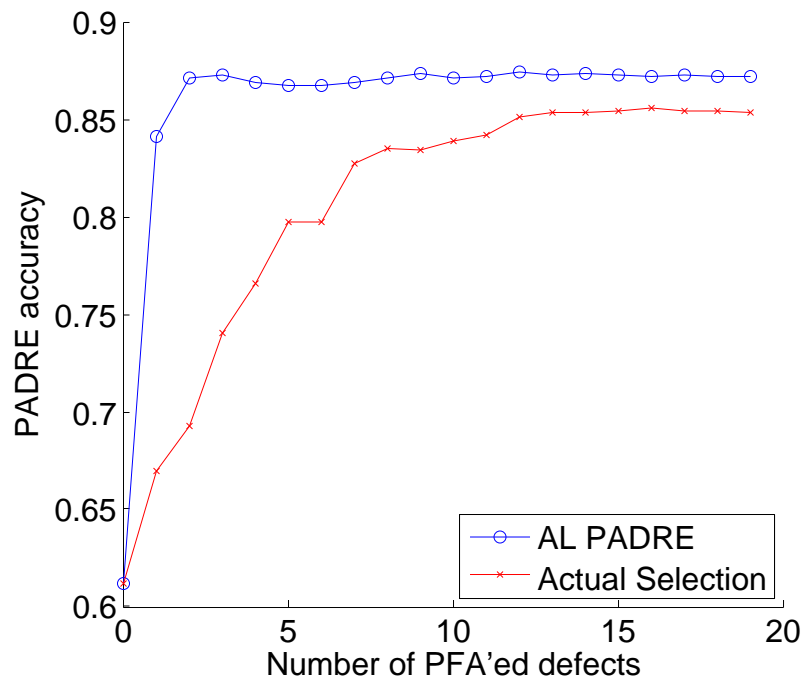


Figure 4.8: PADRE accuracy versus the number of PFA'ed defects for selection using AL PADRE and actual selection made by industry. The initial accuracy is 61.2%.

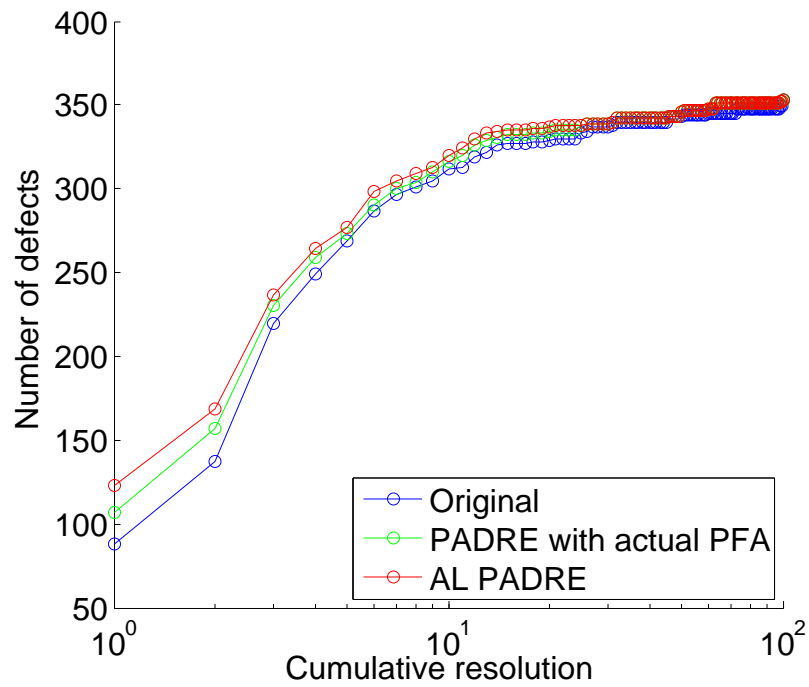


Figure 4.9: The cumulative diagnostic resolution distribution shows PADRE with actual PFA'ed defects increases the number of defects with ideal resolution from 88 to 107, and using AL PADRE can instead improves the number to 123. Each point on the plot shows the number of defects that have a total number of candidates no greater than the given cumulative resolution.

Chapter 5

Changing Failure Mechanisms

In the yield learning process, it is possible that the failure mechanisms that cause chips to be the defective chips may change, due to changes (both intentional and unintentional) in the fabrication process [72]. Because PADRE relies on existing diagnosis data to extract training data and to learn the classifier, its performance may be affected by the change in the diagnosis data as a result of the changing failure mechanisms. It is important that PADRE performs consistently for changes in the failure mechanisms, so that resolution improvement is accurately maintained.

There are two key aspects to address this issue. First is to identify the occurrence of the failure mechanisms change, and to evaluate if this change effects the performance of PADRE. Second is to adjust PADRE to update the classifier with the characteristics of the new found failure mechanisms. Initially, it may sound reasonable to update PADRE only when changing failure mechanisms are detected. However, such strategy has a number of drawbacks. First, it relies on an accurate detection of the changing failure mechanisms. Not only that changing failure mechanisms may be gradual, and hard to identify a clear cut-off line for the transition, but also that it is very difficult to accurately track and determine the changing failure mechanisms. Second, PADRE can easily incorporate new data, such as PFA data, to improve its performance. Not updating PADRE continuously is a waste of such advantage. Therefore, in this chapter, we propose some techniques to continuously update PADRE with the latest data, so that it can

perform robustly in case of changing failure mechanisms.

Specifically, we propose three techniques to cope with the changing failure mechanisms. An accuracy tracking technique is used to evaluate the accuracy of PADRE in real time to detect any changes in the failure mechanisms and to determine if PADRE accuracy is degraded. If needed, PADRE is adjusted with a weighting method that allows it to quickly adjust to the new found failure mechanisms. In the early stages of yield learning, there is likely little diagnosis data and an abundance of change in the failure mechanisms. In order to ensure PADRE is stable during such a time, a cross-dataset training technique is introduced to improve the training of PADRE using diagnosis data from a different design that may have completely different layout, technology node and functionality.

5.1 Accuracy Tracking

To evaluate the accuracy of PADRE, the ground truth for each candidate should be known, so that the labels predicted by PADRE can be checked. In other words, the defect locations have to be known in order to label or mark a candidate as correct or incorrect. For the silicon data, ground truth is only available in limited quantities through expensive PFA. However, similar to the *oracle* used in the AL PADRE experiments, the labels of certain candidates can be accurately inferred using heuristics. Specifically, candidates from defects that contain only one candidate is assumed to be *correct*, whereas candidates from defects that contain more than Q , (where Q is sufficiently large) are all assumed to be *incorrect*. Again, this heuristic is based on the assumption that there should be one and only one correct candidate per each defect. In general, a larger Q lowers the error, but a smaller value for Q increases the amount of incorrect-labeled training data. By taking all the defects with a high number of candidates, the incorrect-labeled training set will mostly consist of incorrect candidates. For example, in this thesis we choose $Q = 20$, which bounds the error at 5%, in other words, at most 5% of the candidates in the incorrect-labeled training set will be actually correct candidates.

Upon receipt of a new batch of diagnosis data, candidate labels predicted by PADRE. PADRE accuracy is evaluated by comparing the predicted labels and the pseudo labels generated by the oracle. Whenever PADRE disagrees with the label from the oracle, it is counted as one misprediction. The accuracy is calculated as total number of mispredictions divided by the total candidate count, i.e.,

$$\text{Accuracy} = \frac{\#\text{Misprediction}}{\#\text{Total_candidates}} \quad (5.1)$$

If the failure mechanisms are *unchanging*, it is expected that the accuracy should be stable and consistent. On the other hand, if there is a significant change in the failure mechanisms that renders the existing PADRE inaccurate, an update to the PADRE training data will be necessary to adjust the PADRE classifier to the latest failure mechanisms.

5.2 Changing Failure Mechanisms Training

In order to quickly adjust PADRE to any new failure mechanisms, training data extracted from the latest diagnosis data should be used to retrain the classifier. However, using only the new training data may lead to overfitting of the latest diagnosis data. In addition, entirely abandoning the existing classifier is not prudent, because it is not clear if the changing of the failure mechanisms is just temperate, and thus revert or further evolve in the future. Preserving the current classifier, which is likely trained with significantly more data, provides a more robust and stable performance in the process of adjusting to the changing failure mechanisms.

To incorporate the new training data while also preserving the existing training data, a weighting strategy is used. Weighting certain training data means duplicating those data multiple times as defined by a weight number w . Weighting is a straightforward strategy to augment under-represented training data in the training [62]. In general, a larger w adapts PADRE more quickly to the latest data, but may lead to overfitting. On the other hand, a smaller value for w is not as efficient in adapting PADRE to the latest data, but it ensures a stable transition.

The weight is identified using cross validation. Cross validation [25] searches through a range of possible weights (\mathbf{w}) for the new training data. For each possible weight w_i , it assesses the accuracy Acc_i that PADRE can achieve using w_i . The weight that provides the highest accuracy is identified as the best after all the possible weights are assessed. To assess the accuracy Acc_i of using each weight w_i , the new training data are divided into ten folds, with each fold j approximately containing an equal number of correct-labeled and incorrect-labeled training candidates, respectively. Then ten experiments are performed. In each experiment j , all nine folds except the j -th fold of new training are weighted by w_i and combined with the existing training data to generate a new classifier ϕ_i^j . Classifier ϕ_i^j is then applied to the j -th fold of the new candidates to evaluate the accuracy using Equation (5.1). After all ten experiments, the accuracy for w_i is calculated as

$$Acc_i = \frac{\sum_{j=1}^{10} Acc_i^j}{10} \quad (5.2)$$

5.3 Cross-Dataset Training

In the early stages of yield learning, there may not be sufficient training data for PADRE to perform well. Instead of waiting for sufficient diagnosis data to accumulate, which delays the yield learning process, it is possible to use training data from other designs as a *prior* for the classifier. By combining the prior and the available new data, PADRE can even be applied at the very early stages of yield learning.

Similar to the challenge of finding the weight when balancing training data for different failure mechanisms, the key is to find the balance between the new data and data for other designs. Again, cross validation is employed to find the weight for the new training data. A ten fold cross validation is used to evaluate the accuracy of the weighted data which consists of training data from other designs and new data weighted with different weights w_i . Again, the accuracy of the weight Acc_i is the average of ten iterations of experiments, as calculated by Equation (5.2).

Our expectation from the early stages of yield learning when the new data are limited, the

classifier will rely more on other design data. But as the new training data accumulates, the classifier becomes more dependent on the new data, allowing the other training data to be phased out without affecting the accuracy of the classifier.

5.4 Experiments

The proposed techniques, namely Accuracy Tracking, Changing Failure Mechanisms Training and Cross-Dataset Training are evaluated with comprehensive simulation experiments. The data used in the simulation experiments are the same data used for AL PADRE, which mimics real failed chips that are affected by various types of defects. The data set contains more than 1,600 defective chips of six different fault types. The chips with injected defects are virtually tested and diagnosed using a commercial flow. The experiments simulate a real production scenario, in which a certain number of failed chips are diagnosed each week, and a small number of them are PFA'ed to identify the root cause of the failure.

5.4.1 Setup

An industrial design is used as the chip under test. The chip used in this experiment is an ASIC manufactured in 130 nm technology that contains about one million gates. The production test set includes 36 scan-chain tests followed by 3,403 logic tests that together achieve 99.5% SSL fault coverage.

More than 1,600 instances of multiple types of faults are randomly injected into the circuit to emulate defective chips. The faults used to emulate the defects are simulated using a commercial tool. The injected faults include: and-, or-, and four-way bridge faults, input pattern faults[66], SSL faults, and MSL faults. The number of injected faults of each type are listed in Table 5.1.

The fault types listed in Table 5.1 are chosen to represent the large variety of actual defects that occur in real chips. The advantage of using a virtual population of failed chips is that it

Table 5.1: Faults Types Injected into the ASIC Design.

Fault type	Number of chips
AND bridge	96
OR bridge	94
Four-way bridge	275
Input pattern	412
SSL	364
MSL	365

provides a large amount of labeled data, *i.e.*, failed chips with known defect locations.

In this section, the diagnosis data are assumed to be streaming in on a weekly basis. Specifically, in each week, 40 failed chips are diagnosed and among which five are selected for PFA. For each week, all the failed chips are of one of the six fault types, as shown in Table 5.2. Because the 40 chips selected each week for the experiment significantly affects the results, each experiment is repeated 300 times to average out the randomness introduced by chip selection.

It should be noted that the setup shown in Table 5.2 is used to simulate a worst-case scenario, in which an abrupt failure mechanisms changing occurs in the production. It is the most challenging scenario for PADRE, because the current training data do not include any data from the new failure mechanisms, and may cause PADRE to mispredict the new data. In a more realistic scenario, a more gradual changing of failure mechanisms is expected, in which PADRE is expected to perform better than in a worst-case, abrupt-changing scenario.

Table 5.2: Faults Types of Failed Chip for Each Week.

Week	Fault type
1-3	SSL
4-6	MSL
7-9	Input pattern
10-12	Four-way bridge
13-14	OR bridge
15-16	AND bridge

In the experiment evaluating the effectiveness of using diagnosis data of a different design as prior, a silicon data set is used; this silicon data set is also used in the PADRE evaluation in

Chapter 3. The silicon data set consists of 5,362 failing circuits, each of which contains 2,309 logic gates. Each circuit has been tested with 5,362 production tests. After initial diagnosis, a total of 17,786 defects and 36,186 candidates are reported. For the second data set, the incorrect-labeled training set is constructed with $Q = 20$ as in the virtual data set experiment. An initial correct-labeled training set with 5,519 candidates and an initial incorrect-labeled training set with 7,376 candidates are constructed.

5.4.2 Accuracy Tracking

Accuracy tracking allows us to evaluate the effectiveness of the existing classifier, which is trained using diagnosis data from previous weeks, to the diagnosis data collected from the current week. The incoming diagnosis data for each week follows the sequence listed in Table 5.2. For instance, in week four, the existing classifier is trained using SSL faults, whereas the new diagnosis data include MSL faults.

In Figure 5.1, the accuracy tracking results for 300 experiments are plotted in the grey box plot and the the mean is plotted in the red curve. As shown in Figure 5.1, the accuracy tracking demonstrates that at the early stage when the training data are limited, the accuracy fluctuates when the defect type changes. This is evident by the accuracy degradation in week four, where the diagnosis data transitions from SSL to MSL. In average, the PADRE accuracy is degraded by 2%. However, as data accumulate, the classifier becomes robust to changing defect types. In addition, the tracking accuracy shows that with a sufficient amount of training data, even fault types that are previously unseen can still be well classified by PADRE. For example, in week thirteen and fifteen, when defect type changes, there is no more accuracy degradation compared to previous week.

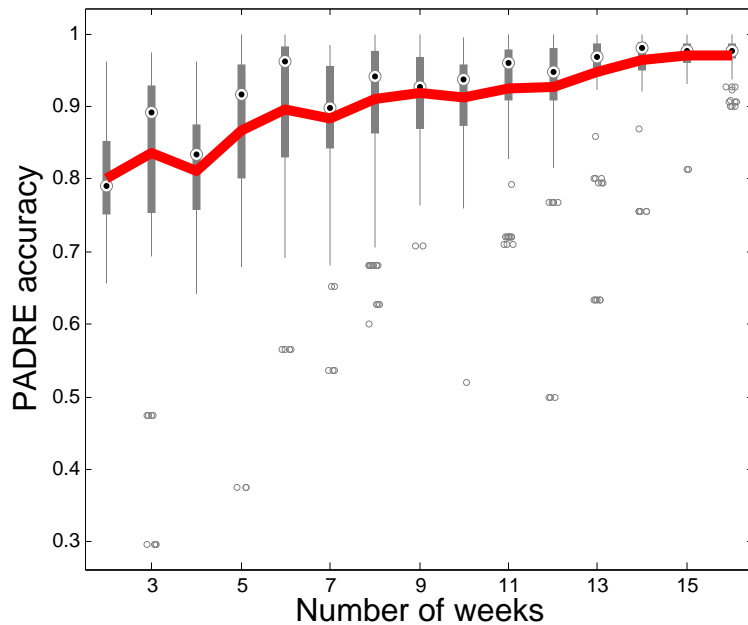


Figure 5.1: The accuracy tracking shows steady increase despite the variance on defect composition (as shown in Table 5.2).

It should be noted that such trend is only visible on a statistical level. For a single experiment, the accuracy can be influenced by other factors, such as the quality of the new diagnosis data received, and the quantity of available data for accuracy verification. This is evident by the large accuracy variance in the early weeks, i.e., early stages of yield learning. It can be observed that, as the diagnosis data accumulate, the accuracy variance decreases gradually, representing more robust performance.

5.4.3 Changing Failure Mechanism Training

To evaluate the impact of weighting on the PADRE prediction accuracy of the data characteristics of Table 5.2, each week the new data are weighted using a weight selected by cross validation. The cross-validated weight for each week in one of the experiments is shown in Table 5.3, and the accuracy of each week is shown in Figure 5.2. In this experiment, the range of weight is from

1 to 100, with a step size of 10, i.e., $w \in \{1, 10, 20, \dots, 100\}$.

As a point of comparison, the same experiments are also repeated with the weight fixed at 20¹. The average accuracy, as the result of the two experiment setups, are compared in Figure 5.3. It can be observed from Figure 5.3 that cross-validated weight provides consistently higher accuracy. On average, the cross-validated weight gives a 2.93% higher accuracy than using a fixed weight. The accuracy advantage is especially significant in the first few weeks, when the accuracy is 4-6% higher using a cross-validated weight over fixed weight.

Table 5.3: Cross-validated weight and accuracy.

Week	Weight	Accuracy
2	50	0.968
3	80	0.992
4	100	0.875
5	10	0.980
6	100	0.950
7	70	0.917
8	50	0.951
9	40	0.965
10	30	0.996
11	60	0.980
12	80	0.947
13	70	0.917
14	100	0.945
15	90	0.927
16	100	0.900

¹This number (20) has been used in previous AL PADRE experiments, it is selected by a trial and error approach for best classification accuracy.

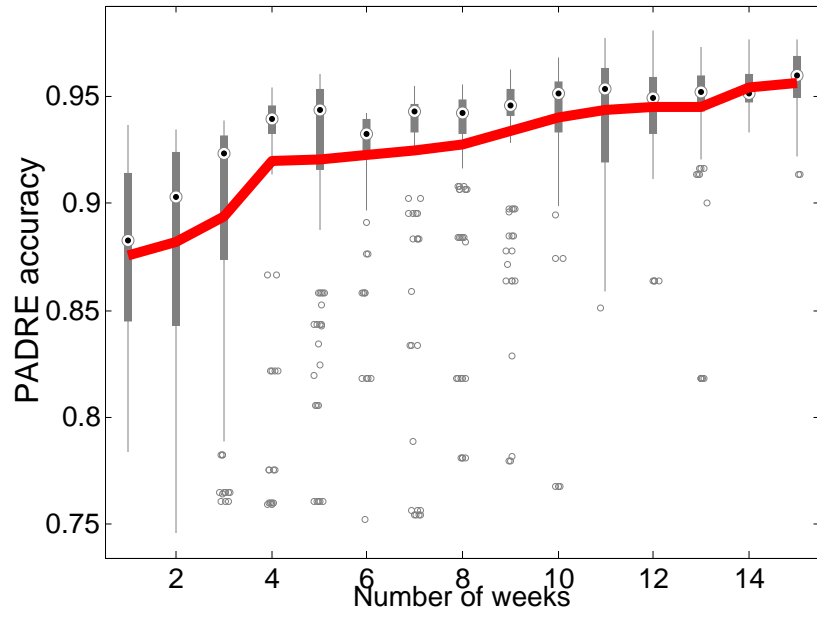


Figure 5.2: The PADRE accuracy of using a cross-validated weight.

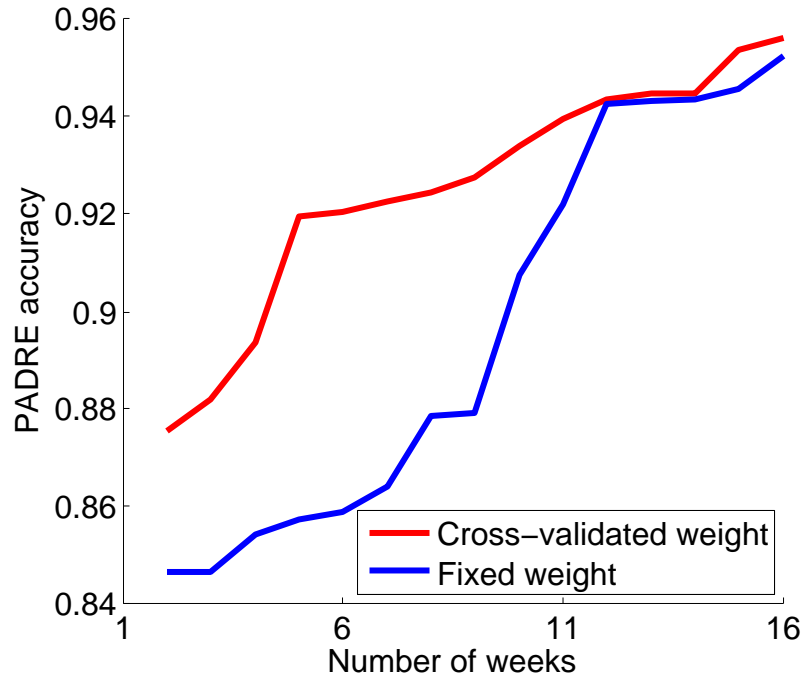


Figure 5.3: The average accuracy using cross-validated weight consistently outperforms that of using a fixed weight of 20.

5.4.4 Cross-Dataset Training

In the evaluation of Cross-Dataset Training, data from a different design is used as prior to boost the PADRE accuracy on targeted data. The silicon data used as the prior consist of 17,786 defects and 36,186 diagnostic candidates. A classifier which is trained entirely using the silicon data, which comprises 5,519 correct-labeled candidates and 7,376 incorrect-labeled candidates, has an accuracy of 81.28% when applied to the simulation data.

The average accuracy of training with a prior is shown in Figure 5.4, and a comparison with the same experiment when a prior is not used is shown in Figure 5.5. It can be seen that using a prior can slightly boost the accuracy compared with the same experiment when a prior is not used. However, it should be noted that the prior itself is not very accurate, with an only 81.28% accuracy. It still provides, however, a modest boost in accuracy even when the classifier has an

accuracy exceeding 90%. If the prior has a better accuracy by itself, for example, if the prior is from the diagnosis data of a similar design, it is expected that the accuracy boost resulted can be more significant.

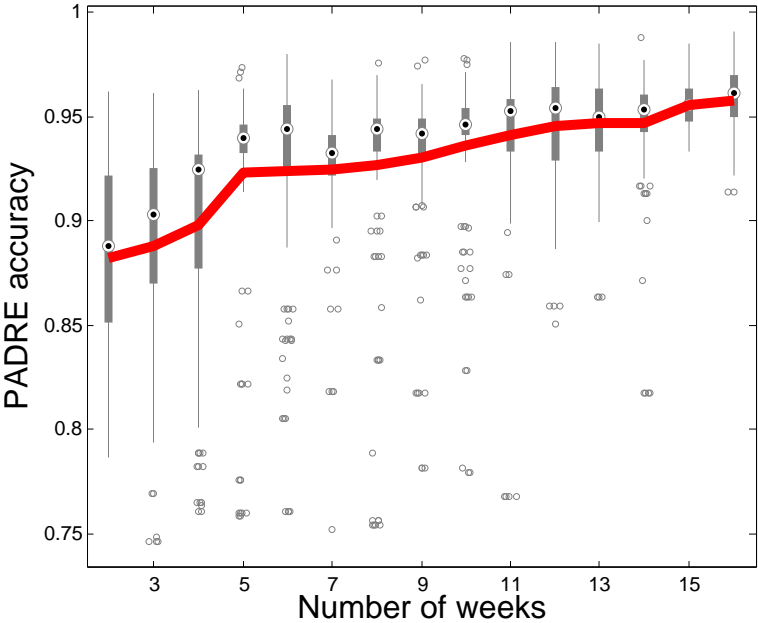


Figure 5.4: The accuracy of using cross-validated weight to combine training data from two different designs.

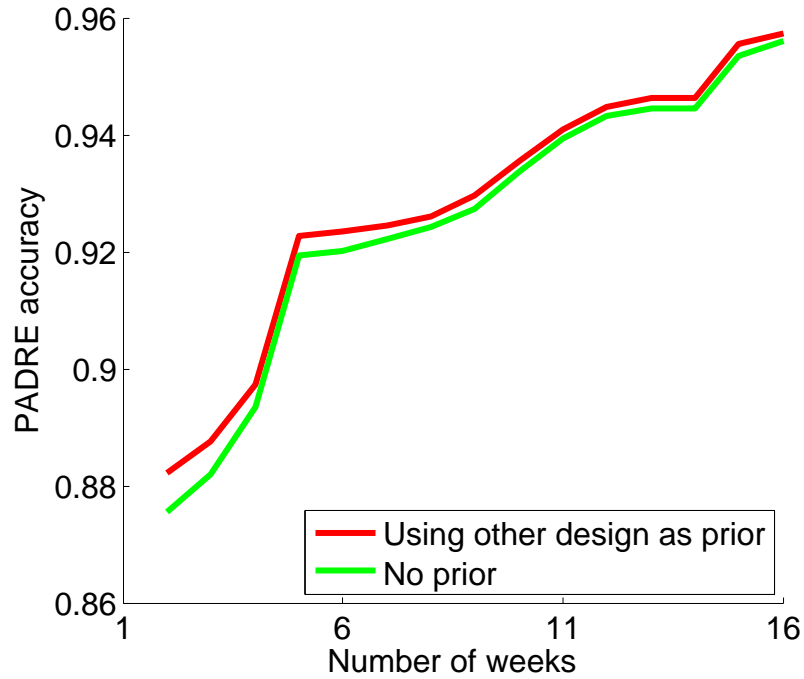


Figure 5.5: The average accuracy with combined training data slightly but consistently outperforms that of using training data from a single design.

5.5 Summary

In this chapter, three techniques are presented, which increase the robustness of PADRE in the face of changing failure mechanisms. Specifically, an accuracy tracking technique is used to identify the changing failure mechanisms and two cross validation-based weighting techniques are used for efficiently and stably adapting PADRE to the latest failure mechanisms. Simulation experiments demonstrate that with the changing failure mechanisms training techniques, the PADRE accuracy is increased by 4-6% at early stages of the yield learning.

Chapter 6

Summary and Future Work

Diagnosis is an indispensable step in the yield learning process. An accurate and high-resolution diagnosis facilitates PFA and other information extraction analysis. Despite the existence of various approaches for improving diagnostic resolution, there is still much room for improvement.

In this work, we present PADRE, a novel machine learning-based resolution improvement technique. With comprehensive study and experiments, we demonstrate PADRE can distinguish the diagnosis candidates by exploiting the logical and physical characteristics derived from the candidates.

6.1 Dissertation Contribution

PADRE, the major contribution of this dissertation, is comprehensively studied and built to effectively improve diagnostic resolution under various production scenarios. It is a software-based technique relying only on easily-available failure diagnosis data, which makes it easy to implement. In addition, it can be easily applied with other diagnosis improvement techniques to further improve accuracy and resolution. The major features and contributions of each aspect of this work are summarized as follows:

PADRE

1. **Training Data Construction.** Unlike other machine learning-based techniques, PADRE constructs correct-labeled and incorrect-labeled training sets from unlabeled candidates that are available, instead of relying on historical data or simulation data. This provides high-quality training data.
2. **Physical Features.** PADRE uses both logical features and physical features to characterize the diagnostic candidates. The physical features characterize not only the candidates themselves, but also the neighborhood condition of the candidates. It is an important addition to the traditional logical features that primarily rely on comparing the pass/fail of the circuit outputs.
3. **Feature Analysis.** A comprehensive study of the features used in PADRE casts light on which features are more effective in identifying the correct candidates, and the features are found to be robust to variations in the defect-type distribution and in the amount of data.

AL PADRE

1. **Resolution-oriented PFA.** AL PADRE selects a defect for PFA with a specific focus on resolution improvement. It is shown that a small number of physical analyses are able to significantly improve diagnostic resolution when defects are carefully selected. AL PADRE can be applied to any pool of potential PFA defects that are identified using other approaches, and it makes a recommendation on which defect should be selected for improving the resolution. It does not require additional physical analysis or testing, but only relies on existing diagnosis data to make the selection, which makes it inexpensive and easy to implement on top of existing selection approaches.
2. **Selection Criteria.** The two selection criteria adopted by AL PADRE combine both classic AL and novel selection criterion that is specially tailored to the characteristics of diagnosis data. It is found that the two criteria are very effective in identifying the defect for PFA for

improving the resolution. In addition, AL PADRE is able to evaluate the effective of its selections for resolution improvement and report when resolution improvement saturates.

3. **Real-Time Application.** AL PADRE can be applied to the real-time yield learning process. It is shown to increase diagnostic resolution in the early stage of yield learning when only a small amount of diagnosis data are available. In addition, the accuracy of PADRE also increases rapidly with the PFA selection made by AL PADRE, especially during the early stage of yield learning when the accuracy is still low due to insufficient data.

Changing Failure Mechanism

1. **Accuracy Tracking.** The novel accuracy tracking method enables a quick, close and reliable watch of the PADRE accuracy given the possible failure mechanisms change. It is shown to identify the performance fluctuation caused by a disruptive failure mechanisms change. In addition, it does not require any additional resources or data than what is used by PADRE, and can be applied at little additional overhead.
2. **Changing Failure Mechanism Training.** A cross-validated weighting method allows PADRE to more quickly adapt to the changing failure mechanisms during the yield learning process. When a change of failure mechanisms causes the existing classifier to be inaccurate, the weighting mechanisms adjusts the training data by assigning a higher weight to the new data. It also avoids overfitting the new data when the changing failure mechanisms are temporal.
3. **Cross-Dataset Training.** To further stabilize and improve the performance of PARDRE in the face of possible failure mechanisms change, diagnosis data from a different design can be used as a prior for training. It is especially helpful at the early stage of the yield learning when there is likely insufficient diagnosis data. Use of other design diagnosis data not only allows PADRE to be applied in the situation of insufficient diagnosis data, but it also improves the PADRE accuracy even after sufficient diagnosis data are accumulated.

6.2 Future Work

Through our work on PADRE, it is apparent that there is always room for improvement for digital diagnosis in terms of its accuracy, resolution, and cost. While PADRE makes some contributions to advancing this continuous endeavor, it also reveals many areas that deserve further study. Here are some of the areas for improving PADRE and digital diagnosis in general.

1. **Defect Characterization.** PADRE focuses on the identification of correct candidates through the analysis of the characteristics of each candidate. However, an important aspect of the candidate is the defect that it is associated with, and the characterization of defects should help the identification of its true candidates, especially when a selection has to be made among a number of candidates that are associated with the same defect.
2. **Defect-based Training.** Through the investigation of Changing Failure Mechanisms Training, it is clear that candidates of different defect types have different characteristics, the extent of such difference deserve further and detailed study. It may even possible to perform more accurate classifications by training multiple classifiers based on defect types, so that the a defect can be classified using the most appropriate classifier.
3. **Cross-techniques Correlation.** Whereas PADRE is very effective in improving the diagnostic resolution, it is still not perfect. There are still many other techniques that directly or indirectly improve resolution. If properly correlated and combined, these different techniques can jointly perform more accurate and high-resolution diagnosis than what is possible by any single technique.
4. **Multiple-instance Learning for PADRE.** PADRE uses innovative heuristics to construct training sets from previously unlabeled candidates. Though effective, the incorrect-labeled training set is inherently *imperfect*, as a small number of correct candidates are knowingly included in the incorrect-labeled training set. Multiple-instance training is a variation of supervised learning algorithm [73]. Unlike typical supervised learning algorithms, which

learn the classifier from two training sets that are made up entirely by incorrect and correct candidates respectively, multiple-instance learning algorithm learns a classifier from a negative training set that contains only correct candidates, and a positive training set that contains *at least* one incorrect candidates [73]. The way multiple-instance training handles the training data appropriately matches the requirement of PADRE. It provides an elegant treatment for the incorrect training set which in fact also contains a small number of correct candidates. Therefore, it is worth exploring as an alternative to the current second-level classifier which uses conventional SVM for the identification of correct candidates.

5. **PADRE Benchmarking.** As reviewed in Chapter 1, there exist other resolution improvement techniques, such as candidate-scoring, diagnostic ATPG, and other machine learning-based approaches [11, 12, 16, 17, 18, 21, 22]. Compared to these resolution improvement techniques in the literature, PADRE has the advantage of being an easy-to-implement approach that does not require additional testing. In addition, compared to other machine learning-based techniques, PADRE adopts a novel approach to generate training data. Specifically, PADRE does not rely on historical diagnosis or simulation data to obtain high-quality training data. However, to comprehensively evaluate PADRE, its performance should be benchmarked against other techniques. The benchmarking is not performed in this thesis for two reasons. Firstly, some of the techniques in the literature are difficult to implement with the available data. For example, RCD [22] requires the detailed information on layout and manufacturing process. Secondly, most conventional techniques are complementary to, instead of competing with, PADRE. In other words, PADRE can be applied in conjunction with other techniques improving diagnostic resolution. For example, it can be integrated with the diagnostic ATPG techniques such as [17, 18] by applying PADRE before the fault isolation step to reduce the number of faults that have to be distinguished in the fault-distinguishing test pattern generation step. Nevertheless, benchmarking is still an important evaluation, since it casts light on how well PADRE can work

with other techniques to jointly improve resolution, which will be considered in our future research.

Bibliography

- [1] Semiconductor Industry Association, “The international technology roadmap for semiconductors 2.0,”
- [2] M. Bushnell and V. Agrawal, *Essentials of Electronics Testing*.
- [3] R. Desineni, O. Poku, and R. D. Blanton, “A logic diagnosis methodology of accurate defect behavior,” *Proc. IEEE International Test Conference*, pp. 1–10, 2006.
- [4] K. De and A. Gunda, “Failure analysis for full-scan circuits,” *Proc. IEEE International Test Conference*, pp. 636–645, 1995.
- [5] L. Wagner, *Failure Analysis of Integrated Circuits Tools and Techniques*, ch. Chemical Analysis, pp. 195–204. Kluwer Academic Publishers, 1999.
- [6] L. M. Huisman, M. Kassab, and L. Pastel, “Data mining integrated circuit fails with fail commonalities,” *Proc. IEEE International Test Conference*, pp. 661–668, 2004.
- [7] W. C. Tam, O. Poku, and R. D. Blanton, “Precise failure localization using automated layout analysis of diagnosis candidates,” *Proc. ACM/IEEE Design Automation Conference*, pp. 367–372, 2008.
- [8] R. D. Blanton and Y. Lin, “Test effectiveness evaluation through analysis of readily-available tester data,” *Proc. IEEE International Test Conference*, pp. 1–10, 2009.
- [9] X. Yu and R. D. Blanton, “Estimating defect-type distributions through volume diagnosis and defect behavior attribution,” *Proc. IEEE International Test Conference*, pp. 1–10, 2009.

- [10] R. C. Aitken, "Finding defects with fault models," *Proc. IEEE International Test Conference*, pp. 498–505, 1995.
- [11] J. A. Waicukauski and E. Lindbloom, "Failure diagnosis of structured VLSI," *IEEE Design and Test of Computer*, vol. 6, no. 4, pp. 49–60, 1989.
- [12] D. B. Lavo, I. Hartanto, and T. Larrabee, "Multiplets, models, and the search for meaning: Improving per-test fault diagnosis," *Proc. IEEE International Test Conference*, pp. 250–259, 2002.
- [13] 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 and Test of Computer*, vol. 29, no. 1, pp. 36–47, 2012.
- [14] R. Desineni and R. D. Blanton, "Diagnosis of arbitrary defects using neighborhood function extraction," *Proc. IEEE VLSI Test Symposium*, pp. 366–373, 2005.
- [15] 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.
- [16] F. Hapke, W. Redemund, A. Glowatz, J. Rajski, M. Reese, M. Hustava, M. Keim, J. Schloeffel, and A. Fast, "Cell-aware test," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 9, pp. 1396–1409, 2014.
- [17] T. Bartenstein, "Fault distinguishing pattern generation," *Proc. IEEE International Test Conference*, pp. 820–828, 2000.
- [18] N. K. Bhatti and R. Blanton, "Diagnostic test generation for arbitrary faults," *Proc. IEEE International Test Conference*, pp. 1–9, 2006.
- [19] J. E. Nelson, W. C. Tam, and R. D. Blanton, "Automatic classification of bridge defects," *Proc. IEEE International Test Conference*, pp. 1–10, 2010.
- [20] S. Wang and W. Wei, "Machine learning-based volume diagnosis," *Proc. Design, Automa-*

tion and Test in Europe Conference Exhibition, pp. 902–905, 2009.

- [21] X. Ren, M. Martin, and R. D. Blanton, “Improving accuracy of on-chip diagnosis via incremental learning,” *Proc. IEEE VLSI Test Symposium*, pp. 1–6, 2015.
- [22] Y. Huang, W. Yang, and W. Cheng, “Advancements in diagnosis driven yield analysis (DDYA): A survey of state-of-the-art scan diagnosis and yield analysis technologies,” *Proc. IEEE European Test Symposium*, pp. 1–10, 2015.
- [23] R. D. Blanton, F. Wang, C. Xue, P. Nag, Y. Xue, and X. Li, “DREAMS: DFM rule Evaluation using manufactured silicon,” *Proc. ACM/IEEE International Conference on Computer-Aided Design*, pp. 99–106, 2013.
- [24] Y. Xue, O. Poku, X. Li, and R. D. Blanton, “PADRE: Physically-aware diagnostic resolution enhancement,” *Proc. IEEE International Test Conference*, pp. 1–10, 2013.
- [25] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2007.
- [26] S. Tong and E. Chang, “Support vector machine active learning for image retrieval,” *Proc. the ninth ACM international conference on Multimedia*, pp. 107–118, 2001.
- [27] S. Tong and D. Koller, “Support vector machine active learning with applications to text classification,” *The Journal of Machine Learning Research* 2, pp. 45–66, 2002.
- [28] T. P. Mohamed, J. G. Carbonell, and M. K. Ganapathiraju, “Active learning for human protein-protein interaction prediction,” *BMC bioinformatics* 11, 2010.
- [29] M. Abramovici and M. A. Breuer, “Multiple fault diagnosis in combinational circuits based on effect-cause analysis,” *IEEE Transactions on Computers*, vol. C-29, no. 6, pp. 451–460, 1980.
- [30] J. P. Roth, “Diagnosis of automata failures: A calculus and a method,” *IBM Journal of Research and Development*, vol. 10, no. 4, pp. 278–291, 1966.
- [31] R. D. Eldred, “Test routines based on symbolic logic statements,” *Journal of ACM*, vol. 6, no. 9, pp. 33–36, 1959.

- [32] K. C. Y. Mei, "Bridging and stuck-at faults," *IEEE Transactions on Computers*, vol. 23, no. 7, pp. 720–727, 1974.
- [33] J. M. Acken and S. D. Millman, "Accurate modeling and simulation of bridging faults," *Proc. IEEE Custom Integrated Circuits Conference*, pp. 1–4, 1991.
- [34] C. F. Hawkins, "Defect classes - an overdue paradigm for CMOS IC testing," *Proc. IEEE International Test Conference*, pp. 413–425, 1994.
- [35] R. C. Aitken and P. C. Maxwell, "Better models or better algorithms? Techniques to improve fault diagnosis," *Hewlett-Packard Journal*, pp. 110–116, Feb 1995.
- [36] R. L. Wadsack, "Fault modeling and logic simulation of CMOS and MOS integrated circuits," *Bell System Technical Journal*, vol. 57, no. 5, pp. 1449–1474, 1978.
- [37] T. M. Storey and J. W. Barry, "Delay test simulation," *Proc. the Design Automation Conference*, pp. 492–494, 1977.
- [38] J. D. Lesser and J. J. Shedletsky, "An experimental delay test generator for LSI logic," *IEEE Transactions on Computers*, vol. C-29, no. 3, pp. 235–248, 1980.
- [39] G. L. Smith, "A model for delay faults based upon paths," *Proc. the International Test Conference*, pp. 342–349, 1985.
- [40] Y. Sato, "A persistent diagnostic technique for unstable defects," *Proc. the International Test Conference*, pp. 242–249, 2002.
- [41] I. Yamazaki, "An approach to improve the resolution of defect-based diagnosis," *Proc. the Asian Test Symposium*, pp. 123–128, 2001.
- [42] P. C. Maxwell and R. C. Aitken, "Biased voting: A method for simulating CMOS bridging faults in the presence of variable gate logic thresholds," *Proc. IEEE International Test Conference*, pp. 63–72, 1993.
- [43] S. Venkataraman and S. B. Drummonds, "POIROT: A logic fault diagnosis tool and its applications," *Proc. IEEE International Test Conference*, pp. 253–262, 2000.

- [44] B. Chess, "Diagnosis of realistic bridging faults with single stuck-at information," *Proc. the International Conference on Computer-Aided Design*, pp. 185–192, 1995.
- [45] S. D. Millman, E. J. McCluskey, and J. M. Acken, "Diagnosing cmos bridging faults with stuckat fault dictionaries," *Proc. the International Test Conference*, pp. 860–870, 1990.
- [46] S. Y. Huang, "On improving the accuracy of multiple defect diagnosis," *Proc. the VLSI Test Symposium*, pp. 34–39, 2001.
- [47] A. Veneris, "Incremental diagnosis and correction of multiple faults and errors," *Proc. the Design, Automation and Test in Europe*, pp. 716–721, 2002.
- [48] V. Bopanna, "Multiple error diagnosis based on xlists," *Proc. the Design Automation Conference*, pp. 100–110, 1999.
- [49] P. Camurati, "A diagnostic test pattern generation algorithm," *Proc. the International Test Conference*, pp. 52–58, 1990.
- [50] J. M. Galey, R. E. Norby, and J. P. Roth, "Techniques for the diagnosis of switching circuit failures," *IEEE Transactions on Communications and Electronics*, vol. 83, no. 74, pp. 509–514, 1964.
- [51] M. Marzouki, J. Laurent, and B. Courtois, "Coupling electron-beam probing with knowledge based fault localization," *Proc. IEEE International Test Conference*, p. 238, 1991.
- [52] I. Pomeranz and S. M. Reddy, "On the generation of small dictionaries for fault location," *Proc. the International Test Conference*, pp. 272–279, 1992.
- [53] M. Abramovici, P. R. Menon, and D. T. Miller, "Critical path tracing - an alternative to fault simulation," *IEEE Design and Test of Computers*, vol. 1, pp. 83–93, 1984.
- [54] S. B. Drummonds, "Bridging the gap between logical diagnosis and physical analysis," *IEEE International Workshop on Defect Based Testing*, pp. 272–279, 2002.
- [55] P. G. Ryan, S. Rawat, and W. K. Fuchs, "Two-stage fault location," *Proc. the International Test Conference*, pp. 963–968, 1991.

- [56] X. Fan, "A novel stuck-at based method for transistor stuck-open fault diagnosis," *Proc. the International Test Conference*, pp. 182–187, 2005.
- [57] R. W. Allen, M. M. Ervin-Willis, and R. E. Tulloss, "DORA: CAD interface to automatic diagnostics," *Proc. the Design Automation Conference*, pp. 559–565, 1982.
- [58] T. J. Vogels, W. Maly, and S. Blanton, "Progressive bridge identification," *Proc. IEEE International Test Conference*, pp. 309–318, 2003.
- [59] C. M. Li and E. J. McCluskey, "Diagnosis of sequence-dependent chips," *Proc. the VLSI Test Symposium*, pp. 187–192, 2002.
- [60] T. Bartenstein, D. Heaberlin, L. Huisman, and D. Sliwinski, "Diagnosing combinational logic designs using the single location at-a-time (SLAT) paradigm," *Proc. IEEE International Test Conference*, pp. 287–296, 2001.
- [61] Cadence Design Systems, Inc., *EncounterTM User Guide*, 4.1.5 ed., 2005.
- [62] 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.
- [63] Q. Gu, Z. Li, and J. Han, "Generalized Fisher score for feature selection," *Proc. International Conference on Uncertainty in Artificial Intelligence*, 2011.
- [64] A. Jain and D. Zongker, "Feature selection: evaluation, application, and small sample performance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, pp. 153–158, 1997.
- [65] D. W. Aha and R. L. Bankert, *Learning from Data*, ch. A comparative evaluation of sequential feature selection algorithms, pp. 199–206. Springer New York, 1996.
- [66] R. D. Blanton and J. P. Hayes, "Properties of the input pattern fault model," *Proc. IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp. 372–380, 1997.

- [67] I. W. Tsang, J. T. Kwok, and P. M. Cheung, “Core vector machines: Fast SVM training on very large data sets,” *Journal of Machine Learning Research*, pp. 363–392, 2005.
- [68] S. Hoi, R. Jin, and M. Lyu, “Large-scale text categorization by batch mode active learning,” *Proc. the International Conference on the World Wide Web*, pp. 633–642, 2006.
- [69] D. Lewis and W. Gale, “A sequential algorithm for training text classifiers,” *Proc. the ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3–12, 1994.
- [70] I. Dagan and S. Engelson, “Committee-based sampling for training probabilistic classifiers,” *Proc. the International Conference on Machine Learning*, pp. 150–157, 1995.
- [71] J. Zhu and Z. Ghahramani, “Combining active learning and semisupervised learning using gaussian fields and harmonic functions,” *Proc. the ICML Workshop on the Continuum from Labeled to Unlabeled Data*, pp. 58–65, 2003.
- [72] M. Keim, N. Tamarapalli, H. Tang, M. Sharma, J. Rajsiki, C. Schuermyer, and B. Benware, “A rapid yield learning flow based on production integrated layout-aware diagnosis,” *Proc. IEEE International Test Conference*, pp. 1–10, 2006.
- [73] A. Stuart, I. Tsochantaridis, and T. Hofmann, “Support vector machines for multiple-instance learning,” *Advances in neural information processing systems*, pp. 561–568, 2002.