Learning Enhanced Diagnosis of Logic Circuit Failures

Submitted in partial fulfillment of the requirements for
the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Soumya Mittal

B.Tech., Electronics and Communication Engineering, IIT Roorkee
M.S., Electrical and Computer Engineering, Carnegie Mellon University

Carnegie Mellon University
Pittsburgh, PA
May, 2020

To Maa, Paa, and Dii

Acknowledgements

I would like to thank my advisor, Professor Shawn Blanton, for his continuous support, constant motivation and endless patience over the years. I am extremely grateful for his continued invaluable insights in my research. He is the epitome of an exceptional mentor. His meticulosity, perspicacity, and strive for excellence and success have helped me to grow, both personally and professionally.

I would like to thank Professor Andrzej Strojwas, Professor Diana Marculescu, Dr. Enamul Amyeen, and Dr. Yan Pan for serving on my doctoral dissertation committee, and providing excellent feedback and constructive criticism on my work.

I am grateful to have learned a lot from my supervisors, John Carulli and Yan, during my internship at GlobalFoundries, and Enamul and Dr. Srikanth Venkat Raman, during my internship at Intel. I would also like to thank my former coworkers at GlobalFoundries and Intel for their help and support.

I would like to express my gratitude towards Prof. Xin Li for giving me an opportunity to work with him during my internship at Carnegie Mellon University (prior to PhD) and inspiring me to pursue research.

I would like to acknowledge my friends, and past and present ACTL (Advanced Chip Test Laboratory) colleagues for interesting discussions (research or otherwise) and furthering my academic development.

I would like to recognize the National Science Foundation (NSF), the Semiconductor

Research Corporation (SRC), and the Neil and Jo Bushnell Fellowship in Engineering for their financial support.

Most importantly, none of this would have been possible without the unwavering love, dedication, encouragement and sacrifice of my parents, Dr. Rajesh and Nandini Mittal, and my sister, Pavni Mittal; thank you for believing in me. I am also grateful for the love and support of my brother-in-law, Vibhu Gupta.

Abstract

As semiconductor manufacturing progresses to smaller process nodes, it is becoming increasingly difficult to climb the yield learning curve rapidly. The rate of yield learning dictates the growth and success of the semiconductor industry, and must be accelerated to fulfill competitive time-to-market, time-to-money and time-to-volume requirements.

Software-based diagnosis plays a crucial role in yield learning. Diagnosis comprehends the test response of a failing circuit to determine the location, and sometimes, in addition, characterize the nature of a defect affecting the failing circuit. Besides identifying likely failure mechanisms and increasing the quality of chip testing, the feedback provided by diagnosis is used to select chips for physical failure analysis (PFA). PFA aims to visually examine a chip to characterize a defect, prevent similar defects in the future and, consequently, improve the design and manufacturing of a chip.

However, PFA is often destructive, time- and cost-intensive, and not always successful. Diagnosis, on the other hand, is non-invasive and time- and cost-effective; moreover, it assists PFA and guides yield learning. The advantages of diagnosis, coupled with the diminishing performance of PFA with advancing technology, make it an encouraging facilitator for rapid yield learning.

Therefore, the objective of diagnosis must be logic-level defect characterization to minimize (and ideally eliminate) the need for PFA, and accelerate yield learning. Logic characterization of a defect includes the derivation of its physical location and precise logic behavior.

In this dissertation, a comprehensive diagnosis methodology is developed to actualize the aforementioned objective.

The developed methodology comprises of three methods. LearnX/MD-LearnX is a physically-aware method that employs (a) the X-fault model to avert the elimination of a correct defect candidate and (b) machine learning to build a candidate-ranking model that learns the hidden correlations between the tester response and the defect candidates to pinpoint the correct candidate.

PADLOC, which stands for Physically-Aware Defect LOcalization and Characterization, improves the physical location of a back-end defect (i.e., a defect that affects one or more interconnects and resides outside a standard cell) returned by LearnX/MD-LearnX by partitioning the defective net into physical subnets and identifying the subnets that influence defect excitation. In addition, PADLOC deduces the precise impact of a defect on the circuit functionality by examining its surrounding circuitry.

NOIDA, which stands for NOise-resistant Intra-cell Diagnosis Approach, pinpoints the location of a defect within a failing standard cell implicated by LearnX/MD-LearnX. In contrast to prior work that typically constructs/employs a fault dictionary, NOIDA ascertains the location as well as the behavior of a front-end defect (i.e., a defect that resides inside a standard cell) by monitoring the logical activity of its intra-cell neighborhood. Additionally, NOIDA is resistant to circuit-level noise that may originate from potentially inaccurate transistor-level simulation.

Results from numerous experiments reveal that our diagnosis methodology outperforms state-of-the-art commercial diagnosis. LearnX/MD-LearnX reports fewer defect candidates than commercial diagnosis for 69.4% silicon fail logs without losing accuracy. PADLOC implicates a smaller physical area for a defect for 47.2% silicon fail logs and attains at most 44X improvement. NOIDA reports an ideal diagnosis for 38.0% more front-end defects, when compared to leading-edge commercial diagnosis. In the presence of noise, NOIDA achieves

an ideal diagnosis 7.6X more often.

In summary, this dissertation endeavors to characterize a defect residing in a logic chip in terms of its precise physical location and logic behavior, which, consequently, most likely, enables rapid yield learning. The deployment of machine learning to pinpoint the correct candidate in LearnX/MD-LearnX, and the investigation of the neighborhood of a defect to determine its exact physical location and logic behavior in PADLOC and NOIDA are the novel components of this dissertation, and the reasons for its superiority over the state-of-the-art.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

More than 50 years ago, it was predicted (rather, observed) that the transistor density of an integrated circuit ("chip") would double every year [1]. Ten years later, the prediction, popularly known as the Moore's law, was revised to doubling every two years [2]. Robert Dennard, around the same time, observed that the chip power density remains constant with decreasing transistor sizes [3]. His postulation, widely known as Dennard scaling, has been one of the primary propellants of Moore's law. In conjunction with Dennard scaling, it implies that the energy efficiency (performance per watt) would double even faster than two years.

Moore's law (along with Dennard scaling) has since then become a de facto standard in the semiconductor industry [4–6]. Moore's law has been the driving force behind the technological advancements for semiconductor industry growth. Numerous inventions and breakthroughs have been sustaining Moore's law to advance semiconductor manufacturing, resulting in an increased chip performance and/or improved power efficiency [6–8]. These innovations include (and are not limited to):

- Decreased transistor sizes, where the transistor density (measured in terms of the number of transistors per mm^2), for instance, has proliferated from less than 500 in

the early 1970s [9–11] to more than a 100M in 2019 [12–17].

- Creative transistor structures such as non-planar transistors (e.g., a fin field effect transistor (FinFET) [17–20] and a gate-all-around FET (GAAFET) [21, 22]).
- New materials, such as the use of hafnium instead of silicon dioxide as a transistor gate dielectric in a 45nm process node [23], and using germanium instead of silicon to build a transistor [24–26].
- Three dimensional monolithic and heterogeneous integration [27–31].
- Next-generation lithography strategies such as optical proximity correction (OPC) [32], phase shift mask [33], multiple patterning [34–40], and extreme ultraviolet lithography (EUV) [40–42].
- Novel processor microarchitectures [43–46].

Unfortunately, the increase in the density and complexity of a chip, which causes an increase in its performance (and energy efficiency), is also accompanied by significant manufacturing challenges.

As a consequence, the task of fabricating a working chip has become extremely difficult to accomplish. Human and equipment errors, contaminants, interactions between the design and the manufacturing process, and/or a complication at any fabrication step can introduce a *defect* into a manufactured chip. A defect is an unforeseen physical irregularity in a manufactured chip that may lead to its failure. A *systematic* defect stems from some flaw in the design, deficiency in the manufacturing process, or interactions between the design and the process. A *random* defect, in contrast, arises due to intermittent equipment excursions or contaminants in the fab.

Besides manufacturing defects, poor design, environmental factors such as temperature and humidity, wearout and process variations can render a manufactured chip inadequate.

The proportion of working chips fabricated is called *yield*. Yield loss is further aggravated by diminishing time-to-volume, time-to-market and time-to-money requirements of the semiconductor industry.

Various strategies are used to identify and characterize the sources of yield loss, and consequently, increase yield. The process of rectifying sources of yield loss to improve chip design and manufacturing is called yield learning.

In-line defect inspection [47, 48], for example, optically examines a wafer to identify imperfections in a chip during manufacturing. However, with shrinking technology, its effectiveness to locate a defect is decreasing because it only inspects the wafer surface, which means that only the defects that are visible on the wafer can be identified by an in-line inspection equipment. In addition, not all defects captured by in-line inspection are "killer" defects[1] [49, 50]. Moreover, its use is limited due to its cost, throughput and ineffectiveness to determine the nature of a defect (i.e., whether it is an open, short, etc.).

Specialized test structures such as comb drives, serpentine structures and ring oscillators are custom designed to easily observe specific defect mechanisms [51–53]. They are transparent to failure and can provide quick feedback early in the yield learning phase to make modifications in the manufacturing process. However, such structures are elementary and do not reflect the range of layout patterns used in random logic; hence, their use is limited to evaluating the process parameters in the early stages of yield learning. During volume production, on the other hand, test structures are only deployed in the scribe lines between customer chips, and thus their ability to find defects is restricted.

Bitmap analysis of a memory is yet another way to identify sources of yield loss during the early/intermediate phases of yield learning [48, 54]. A bitmap visually represents the location of the failing bits within a memory. Because of their regular architecture, memories are transparent to failure; the configuration of a bitmap can be investigated to find the root

[1] A killer defect causes a chip to malfunction.

cause of failure. One advantage of using a memory as a yield learning vehicle is that, in contrast to custom test structures, they may undergo all the processing steps in the entire fabrication flow. Even then, however, because of their regularity, memories utilize only the first few process layers. As a result, they are unable to catch defects in the upper metal layers. In addition, due to their regular design, memories contain limited layout patterns, and hence are unfit to monitor the defects resulting from diverse layout geometries typically found in random logic.

An alternative to memory is a logic test chip that is implemented using a standard automated place-and-route tool [55–62]. Short-flow logic test chips, that may contain diverse layout patterns and are fabricated using a subset of metal layers, face a similar drawback as memories – they are ineffective in identifying a source of yield loss that may otherwise be detected if the entire manufacturing flow is employed. Full-flow logic test chips (including legacy designs retrofitted for the latest technology node), on the other hand, utilize all the process layers and thus are more suited to analyze/monitor yield during the later stages of yield ramp.

Finally, manufacturing test is used to pinpoint the sources of yield loss in a logic test chip and the random logic used in a customer chip. Specifically, the knowledge of how a chip fails manufacturing test is used to identify the location and nature of a defect to increase yield. This process, known as software-based diagnosis (or simply, diagnosis), is the first step in failure analysis (FA). Diagnosis offers several advantages (which are discussed in detail later in Section 1.1). For example, diagnosis makes in-line inspection more effective; the outcome of diagnosis can be correlated with the in-line inspection data to discern killer defects.

Additionally, a statistically significant number of diagnoses can be correlated to discover the underlying yield detractor(s). The corresponding chips can then be physically inspected to verify and characterize the yield detractors. As opposed to diagnosis that is expeditious and non-invasive, physical failure analysis (PFA) is time-consuming and destructive. Nev-

ertheless, PFA provides indisputable confirmation of the presence of a defect and explores its underlying physical cause. It aims to provide a critical understanding of how a defect is introduced in the manufacturing process and thus is crucial to yield learning. The feedback obtained from physical failure analysis is used to prevent similar defects in the future, and consequently, amend the design and/or manufacturing process to increase yield.

This dissertation concentrates on employing manufacturing test to identify the sources of yield loss that influence the random logic in a chip (a customer or a logic test chip). In other words, the focus is on logic defect diagnosis, i.e., the analysis and comprehension of the test failure data to determine the location and the behavior of a defect that affects the logic sub-circuit of a chip, to identify logic yield limiters. The role of diagnosis is especially significant during (a) yield ramping, when logic test chips are deployed and/or customer chips are manufactured prematurely; (b) high-volume manufacturing of customer chips for continuous yield learning/monitoring; and (c) yield excursion (i.e., when yield plummets abruptly from its stable value), when diagnosis can be exploited to uncover the underlying source of yield loss and, consequently, stabilize yield.

The next section, Section 1.1 details the role of manufacturing test and diagnosis in failure analysis that, in turn, facilitates yield learning and monitoring. It also lays the groundwork for the work developed in this disseration. Section 1.2 previews the diagnosis methodology developed and highlights the superior performance achieved by our diagnosis methodology when compared to state-of-the-art commercial diagnosis. Section 1.3 presents the organization of the rest of this dissertation.

1.1 Diagnosis Guided Yield Learning

Figure 1.1 illustrates a simplified flow that depicts the role of manufacturing test and diagnosis in yield analysis, learning and monitoring. The flow is partitioned in two parts – the

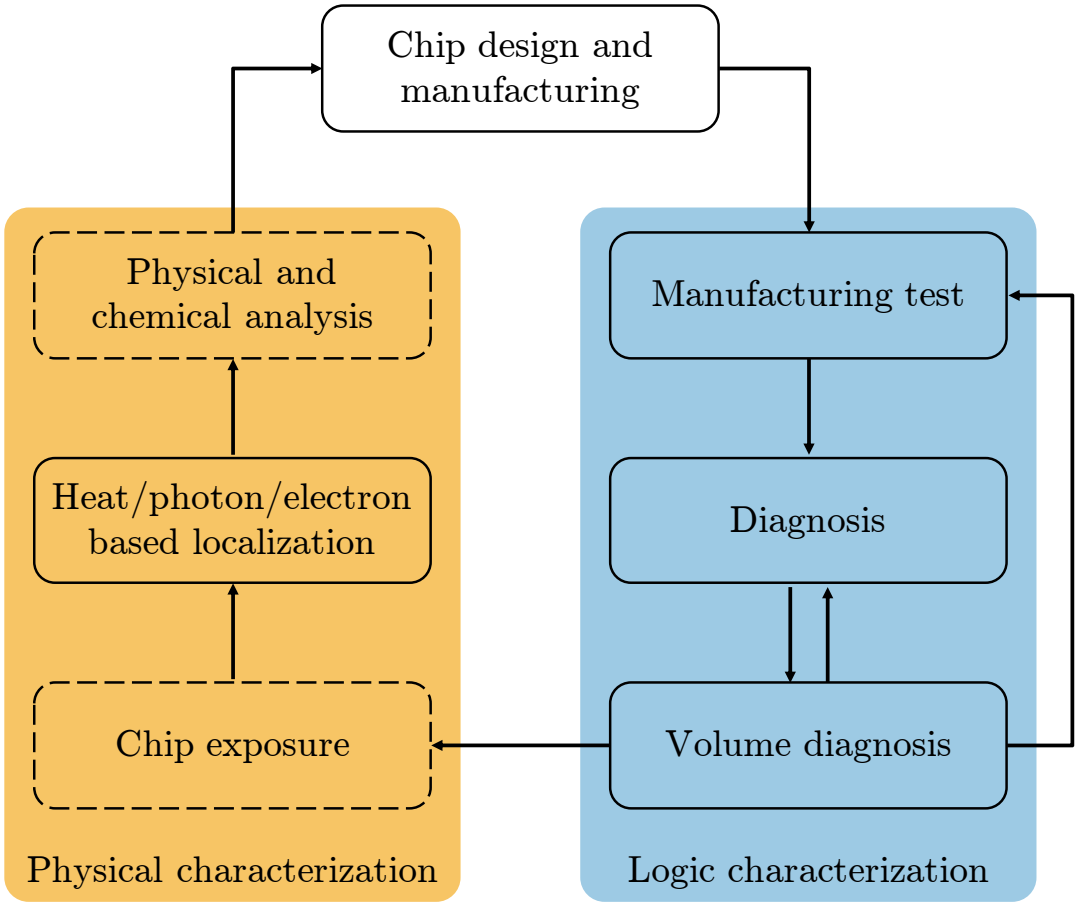


Figure 1.1: A simplified flow depicting the role of manufacturing test and diagnosis in guiding yield analysis, learning and monitoring.

"right" part of the flow (in blue) characterizes a defect at a logic level and the "left" part of the flow focuses on the physical level. While logic characterization focuses on identifying the logic properties of a defect such as its logic behavior and $x - y - z$ location in the circuit, physical characterization aims to uncover the physical features of a defect such as its shape, size, composition and root cause. Each step of the flow depicted with a dashed outline denotes that it is a destructive process. The remaining steps depicted with a solid outline are non-destructive in nature.

The flow begins with the design and manufacturing of a chip. Process-design interac-

tions, design marginalities, and/or complications at any step of the convoluted and precisely regulated fabrication flow can introduce one ore more defects in a produced chip. It should be recalled that the focus of this dissertation is on defects that affect the logic sub-circuit of a chip; thus, the flow of Figure 1.1 will be discussed while considering only defects that affect the logic.

The next step is manufacturing testing[2], where a set of input values called *test patterns* are applied to a logic circuit and the observed circuit response to each pattern is compared with the expected correct response. A circuit that fails testing is classified as defective. Thus, test functions as a binary filter – it is used to differentiate the good chips from the malfunctioning ones. However, testing is not perfect; some good chips can be categorized as defective and vice versa. The fraction of good chips that fail testing is called *test yield loss*. This misclassification is generally due to factors such as a test equipment error and an unsuitable test environment for the circuit under test. The proportion of defective chips that pass testing is termed as *defect level*. This misclassification, where a defective chip escapes test, is due to the inability of a testing approach to detect a defect.

Ideally, a testing technique should detect all possible defects that could render a chip malfunctioning. However, defects are unpredictable, and design- and process-dependent. It is thus impractical to track the sheer number of ways in which a defect may manifest. Testing is made tractable through the use of *fault models*. A fault model is an abstraction from actual defects at a high level. It is a set of assumptions that describes the behavior of a defect and/or its impact on the circuit. Because a fault is defined at a higher level of abstraction than a defect, many defects can map to a single fault. In this way, fault modeling mitigates the complexity of generating a test for each possible defect.

[2]We will focus here on manufacturing testing, which is performed after manufacturing and before chips are shipped to a customer. Other types of testing include characterization testing, which is performed to find design errors and determine the exact operating conditions of the design before fabrication, and acceptance testing, which is carried out by a customer.

Fault models can be defined at different levels of abstraction such as logic (or standard-cell), circuit (or transistor) and physical layout levels. For example, the single stuck line (SSL) fault model (also known as the stuck-at fault model) assumes that a signal in the logic circuit is permanently stuck at either logic-0 or logic-1. The wired-AND (wired-OR) bridge fault model assumes that logic-0 (logic-1) at one of the bridged signals overrides logic-1 (logic-0) at the other bridged signal [63, 64]. The X-fault model assumes an unknown value (X) at a potential defect location [65]. The aforementioned models are defined at the logic level. The transistor stuck-closed (stuck-open) fault model, which is defined at the circuit level, assumes a transistor to be conducting (non-conducting) permanently [66, 67].

A variety of fault models at different abstraction levels with varying complexity exist in the literature to represent ever-evolving defects. Notable examples include the input-pattern [68–70], voting bridge [71, 72], cell-aware [73–78], inductive [79, 80], path-delay [81] and transition [82, 83] fault models.

An alternative approach to increase the quality of a test set is to specify/measure the thoroughness of a test set. For example, the N-detect metric requires at least N detections for each stuck-at fault in the circuit [84,85]. The gate-exhaustive test metric generates a test set such that each standard cell in the circuit is exhaustively tested[3] [86, 87].

After a high-quality test set is generated (while considering different fault models and test metrics) using an Automatic Test Pattern Generation (ATPG) tool, the next step in the yield learning flow (shown in Figure 1.1) is diagnosis. Diagnosis is a software-based process to determine the possible locations (often called candidates) and the corresponding defect types within a failing chip by analyzing and understanding the observed circuit response. The quality of diagnosis is typically measured by the number of candidates identified, the corresponding implicated physical area (i.e., the layout area occupied by the candidates

[3]The gate-exhaustive test metric is equivalent to the input-pattern fault model applied at the standard-cell level.

suspected by diagnosis), and whether the candidate set contains an actual defect location.

Numerous methods have been proposed in the past to enhance the quality of diagnosis. A cause-effect approach compares the tester response with a set of simulated faulty behaviors to find candidates [88–103]. However, because the defect universe is expanding with advancing technology, it is unreasonable to simulate faults that represent all possible defects that could affect a chip. Design size, defect multiplicity and defect behavior variability further restrict the practical applications of a cause-effect approach.

An effect-cause approach, on the other hand, follows an orthogonal approach – it analyzes the observed circuit response to deduce candidates [91, 104–114]. An effect-cause technique such as path tracing [108–110], by itself, however, suffers from poor resolution; because it is cautious about removing a correct candidate, several incorrect candidates are implicated as well.

Methods that include [115–145] use a combination of a cause-effect and an effect-cause approach to benefit from both types of methods. Those techniques usually perform diagnosis in three steps – an effect-cause technique to identify the failing region, simulation of faults within the failing region, and a candidate-scoring procedure to determine possible candidates. Layout information can further be utilized to pinpoint a physical location of a defect residing outside a standard cell [142–161] or within a cell [80, 162–181].

Besides the development of a better algorithm, the effectiveness of diagnosis can also be improved via targeted test generation to distinguish diagnostic candidates [182–212] and/or by altering the design itself [59, 212–216].

The next step in the flow of Figure 1.1 is volume diagnosis, where diagnosis results for a population of failing chips are statistically examined to find yield-limiting design and/or manufacturing issues, in hope of determining if a significant percentage of chips are failing due to a common root cause [144, 217–232]. Chips with yield-limiting defects can then be further inspected to unearth the physical characteristics of a defect.

In addition to assisting/accelerating the subsequent failure analyses, volume diagnosis provides useful information to improve test and diagnosis as well. For example, it can be used to estimate defect-type distribution, which can then be used to reduce test escapes via adaptive testing [233–235]. In [236], diagnosis data is used to find the efficiency of new and existing fault models and test metrics, instead of relying on time-consuming test set silicon experiments. It has also been shown that information obtained from diagnoses can be used to estimate defect density and size distributions (DDSDs) for each metal layer to understand the impact of random defects [237, 238]. In [239], diagnosis is used to predict chip defect level and monitor chip quality. In [240–243], the effectiveness of Design-for-Manufacturability (DFM) guidelines is evaluated using volume diagnosis results.

The steps of Figure 1.1 discussed up to this point enable logic characterization of a defect. The next sequence of steps seeks to physically characterize a defect and investigate its underlying origin.

Based on the output of volume diagnosis, selected chips are further analyzed. The first step towards physical characterization of a defect is exposing the chip, i.e., removing the connections between the chip and its packaging, while ensuring that the chip is electrically functioning. The process of exposing a chip so that it can be inspected visually is called decapsulation (or delidding) [244, 245]. Decapsulation can be accomplished by acid [246], plasma [247, 248], laser [249–251] or thermo-mechanical [252, 253] based methods.

The next step is to isolate the defect location reported by (volume) diagnosis via mechanical, thermal, optical, or electron probing methods. For example, fluorescent microthermal imaging (FMI) studies the temperature-dependent fluorescence emitted by a film coated on the surface of a chip to detect a defect [254–257]. Techniques such as laser-induced voltage alteration (LIVA) [258] and laser-assisted device alternation (LADA) [259] study the interactions of a defect with photons. Other popular probing methods include photon emission microscopy (PEM) [260], optical beam induced current (OBIC) [261], opti-

cal beam induced resistance change (OBIRCH) [262], thermally induced voltage alteration (TIVA) [263], electron beam induced current (EBIC) [264, 265], and charge-induced voltage alteration (CIVA) [266]. In general, those methods are non-destructive in nature, and their goal is to further pinpoint and/or verify the defect location implicated by diagnosis.

Once the presence of a defect is confirmed by non-destructive means, the chip is physically inspected to determine the physical cause of the underlying failure. The investigation begins with a process known as deprocessing (also called delayering) [244], where process layers are carefully removed one at a time until the layer where the defect is suspected is reached. Deprocessing is essentially the reverse of the manufacturing process. An alternative to chip deprocessing is chip cross sectioning, where the chip is grinded and polished perpendicular to its surface to expose all the layers and the transistors simultaneously [244].

Deprocessed or cross-sectioned chips are then inspected via a microscope to observe the physical features of a defect. Defect images can be analyzed, documented and stored to comprehend the physical root cause. Several chip microscopy techniques exist in the literature that aid in the understanding of the physical characteristics of a defect. Some of the popular physical inspection methods include scanning electron microscopy (SEM) [267, 268], scanning probe microscopy (SPM) [269], and transmission electron microscopy (TEM) [268, 270–272]. Chip sample preparation and physical inspection are not necessarily carried out in sequence only once. Defect images obtained from physical inspection can be utilized to guide the process of sample preparation until the defective region of interest is reached. The defective region can further be analyzed via techniques such as energy-dispersive X-ray spectroscopy [273] to chemically characterize the defect, or specifically, identify the atomic structure and composition of a contaminant causing the defect.

Although physical failure analysis (PFA), i.e., the collective process of preparing the chip sample, and physically and chemically inspecting the prepared sample for defect characterization, is a time-consuming, meticulous and invasive process, it provides irrefutable evidence

of the existence of a defect. In addition, the knowledge of the physical and chemical properties of the defect can suggest useful information regarding its origin or root cause. Corrective actions can then be taken to adjust the design or the manufacturing process to increase yield.

High diagnosis quality is thus extremely important for improving design, test and manufacturing of a chip. An inaccurate diagnosis, for example, can direct volume diagnosis to find false correlations within diagnosis results, which, in turn, can steer PFA to physically inspect incorrect locations. The destructive and resource-intensive nature of PFA constrains it to being performed on only a small number of chips and considerable amount of resources can be wasted if diagnosis is inadequate. Additionally, PFA is becoming challenging because of increasing chip density and decreasing critical defect size with advancing technology. Therefore, a highly effective diagnosis methodology makes PFA more successful, and more cost- and time-efficient.

To summarize, diagnosis is an invaluable engine that critically impacts the

- derivation of DDSDs for each metal layer to understand the effect of random defects on yield,
- construction of a pareto of likely failure mechanisms that decrease yield,
- quality of an adaptive test set that reduces test escapes,
- comprehension of the relative performance of new and existing fault models and test metrics to optimize test quality and cost, and
- success rate and the efficiency of PFA.

As a consequence, diagnosis, in all likelihood, accelerates/facilitates yield analysis, learning and monitoring.

1.2 Dissertation Overview

Determining the location of a defect is the traditional objective of diagnosis. Ideally, diagnosis should go beyond localization to characterizing both the nature and root cause of a defect. Because diagnosis is non-invasive, has a quick turnaround time, is becoming useful for other applications, and, more importantly, plays a vital role in aiding yield analysis, learning and monitoring (illustrated in Figure 1.1 and described in Section 1.1), it is imperative that diagnosis continue to improve and move toward characterization.

Therefore, the goal of diagnosis should be twofold.

1. Accurately pinpoint the $x - y - z$ layout location of a defect.

2. Precisely derive the logic behavior of a defect, i.e., deduce the precise impact of a defect on circuit behavior (with minimal assumptions), and identify the defect type.

The accomplishment of the aforementioned two objectives of diagnosis accelerates the overall flow of Figure 1.1. For example, the onus of characterization of a defect that is typically on cost- and time-intensive PFA can be shared by cost- and time-effective diagnosis that adheres to the above-mentioned goals. An ideal diagnosis would thus minimize the need for PFA, thereby, further enabling rapid yield analysis.

Numerous methods have been recommended in the literature to enhance the quality of diagnosis. Diagnosis quality can essentially be improved in three ways – better algorithms, better tests and/or better design. Diagnosis methods that develop new algorithms can further be differentiated based on the type of fault model exploited at the logic level (whether a binary error or an unknown value is assumed at a candidate location), the scoring technique employed (deterministic vs. statistical), how precisely a defect is localized (i.e., whether a defect candidate is reported at a logic, back-end layout or front-end layout level), and whether multiple defects affecting a single chip can be analyzed/identified adequately.

To the best of our knowledge, no prior diagnosis methodology simultaneously focuses on the two objectives of an ideal diagnosis method, i.e., accurate defect localization at a physical-level as well as precise defect behavior identification at a logic-level.

A comprehensive diagnosis methodology is presented in this dissertation that attempts to accomplish each of the two goals of diagnosis. The focus is on improving the quality of diagnosis via designing a better algorithm; approaches that manipulate a test set or alter the design itself complement and strengthen our work [59, 182–216].

Our developed methodology can be divided into three stages. Specifically,

- LearnX [274] is a physically-aware diagnosis methodology that focuses on accurate defect localization. Besides employing the widely used (temporary) stuck-at fault model to identify "simple" defects, LearnX takes the advantage of the X-fault model [65] to locate a defect that misbehaves arbitrarily. The X-fault model allows an error to propagate conservatively and thus (likely) precludes the elimination of a correct candidate.

  Additionally, instead of manually devising a candidate ranking procedure by intuition and/or domain knowledge that may or may not work for ever-changing defects, machine learning is exploited to construct a data-driven ranking model that learns the hidden correlations between the observed circuit response and the (fault simulation response of the) correct candidate. Moreover, it utilizes design layout information to find the candidates that are physically feasible to identify their defect types.

  Simulation-based experiments conducted for four different designs demonstrate the potential of LearnX. LearnX identifies the correct candidate for 97.3% fail logs, which is 10.5% better than leading-edge commercial diagnosis. In addition, LearnX returns an ideal diagnosis (i.e., a single correct candidate) significantly more often than commercial diagnosis. Specifically, LearnX reports an ideal diagnosis for 67.9% fail logs, which

is 67.5% better than one commercial tool and almost double the second commercial tool.

Moreover, significance of LearnX is affirmed by diagnosing 2,400 silicon fail logs from a design fabricated in an advanced process technology. It is revealed that LearnX returns an ideal resolution for 46.9% fail logs, which is almost twice more than what commercial diagnosis achieves. Additionally, LearnX returns 7.1 fewer candidates per fail log, on average.

The effectiveness of LearnX is further corroborated by inspecting 19 failing chips that are PFA'ed. LearnX is able to correctly locate a defect in each failing chip, while reporting fewer candidates than state-of-the-art commercial diagnosis. It is observed that LearnX returns an ideal diagnosis for 52.9% fail logs, which is 12.6% better than commercial diagnosis.

The experiment results thus far underscore the performance of LearnX when a single defect affects a failing chip. A diagnosis methodology called MD-LearnX [275], which builds on LearnX, is developed to effectively tackle multiple defects in a failing chip. MD-LearnX employs the X-fault model as well to avoid the effects of error masking and unmasking that are prominent due to the interaction of multiple defects, and deploys machine learning to identify the best candidate for each defect.

A thorough simulation-based experiment is conducted to assess the capability of MD-LearnX to identify multiple defects, where a total of 28,000 faulty circuits with varying defect multiplicities and behaviors are created and analyzed. Three metrics, namely, diagnosability (i.e., proportion of injected defects that are correctly located), precision (i.e., proportion of reported defects that are correctly identified) and *home run* (i.e., when a single correct candidate is reported for each injected defect), are employed to measure the performance of MD-LearnX.

It is observed that MD-LearnX achieves an average diagnosability and precision of 0.78 and 0.7, respectively, which is 15.4% and 57.0% better than leading-edge commercial diagnosis. MD-LearnX delivers a home run for 40.0% of fail logs, which is twice as often as commercial diagnosis.

The efficacy of MD-LearnX is impressive for large values of defect multiplicity. Specifically, when the number of injected defects is at least five, the diagnosability of MD-LearnX is 22.8% higher than commercial diagnosis, on average. MD-LearnX returns a correct candidate for each reported defect 2.4X more often than commercial diagnosis. Moreover, MD-LearnX delivers a home run for 6.8% fail logs; however, commercial diagnosis returns an ideal diagnosis for less than 0.3% of fail logs.

The capability of MD-LearnX to diagnose multiple defects is further demonstrated with a silicon experiment, where 17 failing chips that are suspected to be affected by multiple defects are diagnosed. It is seen that MD-LearnX returns fewer candidates than commercial diagnosis for 88.2% of the fail logs, without sacrificing accuracy. Moreover, MD-LearnX reports 8.5 fewer candidates per fail log, on average.

- PADLOC, which stands for Physically-Aware Defect LOcalization and Characterization, specializes in accurate physical localization and behavior derivation of a back-end defect (i.e., a defect that affects one or more interconnects and resides outside a standard cell) [146]. It analyzes the back-end defect candidates reported by LearnX/MD-LearnX. PADLOC partitions a candidate net into one or more segments by analyzing the topology of the net and its physical neighbors, and examines the logical activity on the adjacent segments to deduce defect excitation conditions.

  Results from a comprehensive simulation-based experiment reveal that PADLOC pinpoints a defect accurately for 96.5% fail logs, which is 10.3% better than state-of-the-art commercial diagnosis. More impressive, however, is the physical resolution attained by

PADLOC. It is observed that the average *bounding circle diameter*, which is defined as the diameter of the smallest circle enclosing the suspected layout regions, for PADLOC is 61.2, and is 33.0% better than commercial diagnosis.

A silicon experiment is also conducted to demonstrate the capability of PADLOC. Silicon failure data corresponding to 2,400 failing chips fabricated in an advanced process technology is examined. The analysis reveals that PADLOC reports a smaller bounding circle for 68.8% of fail logs, when compared with commercial diagnosis. In addition, the *bounding circle diameter* for PADLOC is 120.3, on average, which is 32.7% smaller (i.e., better) than commercial diagnosis.

PADLOC is further validated by inspecting 36 failing chips that are PFA'ed. PADLOC is able to physically locate a defect in each failing chip correctly, while suspecting a smaller layout region than state-of-the-art commercial diagnosis. It is seen that PADLOC improves *bounding circle diameter* for 47.2% of fail logs, and achieves up to 44X improvement.

- NOIDA (NOise-resistant Intra-cell Diagnosis Approach) concentrates on diagnosing a front-end defect, i.e., a defect that resides inside a standard cell [162]. It analyzes the standard cell candidates reported by LearnX/MD-LearnX. Rather than relying on an existing fault model or creating a new model via potentially inaccurate SPICE models, NOIDA derives the precise impact of a defect within the standard cell by monitoring the logical activity of its physical intra-cell neighborhood. The output of NOIDA is a set of intra-cell candidates, where each candidate is characterized with respect to its physical location inside the cell and its likely behavior.

  A comprehensive simulation experiment is conducted to study the effectiveness of NOIDA. NOIDA is evaluated on over 34,000 front-end defects that are distributed over 115 standard cells within a 7nm standard-cell library.

When each defective cell is exhaustively tested, NOIDA accurately diagnoses 96.6% of defects, which is an improvement of 17.7% over state-of-the-art commercial diagnosis. In addition, NOIDA reports an ideal diagnosis for 50.5% more defects.

When each defective cell is tested with a test set generated by a commercial ATPG tool, it is observed that the average accuracy for NOIDA is 100.0%, which is 21.8% better than commercial diagnosis. Among the defects diagnosed with a single candidate, NOIDA is correct for 38.0% more fail logs.

More importantly, NOIDA is robust to circuit-level noise (that might stem from approximate SPICE modeling) compared to leading-edge commercial diagnosis. Experiment results indicate that NOIDA correctly diagnoses 78.5% of the front-end defects, which is more than twice as often as commercial diagnosis. NOIDA returns an ideal diagnosis for 13.2% defects, which is 7.6X times commercial diagnosis.

1.3 Dissertation Organization

The rest of the dissertation is organized as follows. Chapter 2 describes LearnX, a physically-aware diagnosis methodology that concentrates on defect localization. It discusses LearnX while assuming that a single defect is residing in a failing chip. Chapter 3 extends that discussion to multiple defects, and presents a diagnosis methodology called MD-LearnX. Chapter 4 sheds light on PADLOC (Physically-Aware Defect Localization and Characterization), which specializes in characterizing a back-end defect candidate reported by LearnX/MD-LearnX with respect to its physical location and logic behavior. Chapter 5 elucidates NOIDA (Noise-resistant Intra-cell Diagnosis Approach), which pinpoints a physical location and derives the behavior of a defect candidate within a standard cell that is suspected to be failing as reported by LearnX/MD-LearnX. Each chapter critiques related prior work, and motivates the need for the development of the corresponding diagnosis methodology. The superior

performance attained by each diagnosis method over leading-edge commercial diagnosis is demonstrated with extensive simulation-based and silicon experiments. Finally, Chapter 6 emphasizes the main contributions of our work and provides promising directions for future work.

Chapter 2

LearnX: A Deterministic-Statistical Single Defect Diagnosis Methodology

The goal of software-based diagnosis is to determine the location, and ideally, the precise logic behavior of a defect. This goal is accomplished in multiple stages in this dissertation. This chapter focuses on the first stage of diagnosis, which is defect localization. This chapter describes a two-phase, physically-aware diagnosis method called LearnX to effectively localize a defect (and determine its type) in a failing chip.

The first stage of diagnosis, in particular, is significant because it affects the performance of the subsequent stages. The output of LearnX is used as an input to the back-end and the front-end layout analysis techniques (called PADLOC [146] and NOIDA [162], respectively) developed in this dissertation. As a result, any loss in the accuracy of LearnX directly impacts the effectiveness of PADLOC and NOIDA. Thus, the quality of defect localization achieved by LearnX is crucial to the success of our overall diagnosis methodology.

As alluded to earlier, LearnX is two-phase diagnosis approach. The first phase attempts to diagnose a defect that manifests as a well-established fault behavior (e.g., the stuck-at or bridge fault models). The second phase uses machine learning to build a model (separate for

each defect type) that learns the characteristics of defect candidates to distinguish correct candidates from incorrect ones.

The rest of the chapter is organized as follows. Section 2.1 reviews prior work in the area of diagnosis (specifically, defect localization[1]), and motivates LearnX. Section 2.2 provides a detailed overview of the two phases of LearnX to identify a defect in a failing chip. The efficacy of LearnX is demonstrated via several experiments, which are described in Section 2.3. Finally, Section 2.4 summarizes the discussion on LearnX.

2.1 Prior Work

Numerous algorithms have been proposed over the years to improve the diagnosability of a failed circuit. Diagnosis algorithms can be broadly categorized into cause-effect and effect-cause techniques. Cause-effect based approaches build a database of simulated faulty behaviors and compare them with the observed tester response [88–103]. The database containing fault simulation responses is typically referred to as a fault dictionary. A primary advantage of employing such an approach is that it characterizes a defect with respect to its location and logic behavior. However, a fault dictionary based method is limited by at least the following three factors, and is hence impractical (if used, by itself, on the entire design).

- The size of the dictionary, which is proportional to the size of the design and the number of fault models employed.
- The compute time to construct the dictionary, where each targeted fault needs to be simulated.
- The fidelity of the selected fault models.

[1]Prior work related to the characterization of a back- and a front-end defect is scrutinized in Section 4.1 and Section 5.2, respectively.

Although the approaches presented in [91–103] can reduce the memory requirements of storing the dictionary by compressing the fault simulation responses, and the approaches presented in [276, 277] can reduce the fault simulation time via graphic processing units (GPUs), the practicality of a cause-effect method heavily relies on the accuracy of the chosen fault models. Smaller process nodes increase manufacturing complexity and chip density, which results in new defect types and failure mechanisms. As a consequence, the detection, let alone diagnosis, of a defect becomes challenging. Furthermore, a cause-effect method, by design, is incapable of effectively diagnosing an unmodeled defect.

An effect-cause technique, on the other hand, adopts an orthogonal path. An effect-cause technique analyzes the faulty behavior, and deduces one or more defects that adequately explains the observed tester response [91, 104–114]. Popular effect-cause techniques include back-coning [278] and path tracing [108–110]. Back-coning analyzes the logic fan-in cone of each failing output to identify likely failing regions in a circuit. Path tracing is an upgraded version of *critical path tracing* [111–113] and the algorithm developed in [114]. The techniques of [111–114] are however restricted to single stuck-at faults, i.e., they assume a single signal in the design to be faulty. Path tracing follows a conservative approach and is guaranteed to include all signals which could be defective, even for multiple defects.

The following rules are followed during path tracing.

- If a cell output is reached and all the cell inputs are non-controlling, tracing continues from each input.
- If a cell output is reached and one or more cell inputs are controlling, tracing continues from all the controlling inputs.
- If a fan-out branch is reached, tracing continues from the stem.

The example circuit of Figure 2.1 is used to illustrate back-coning and path tracing. The figure shows the state of the circuit for a failing pattern, $p(abcde) = 00111$. The value at

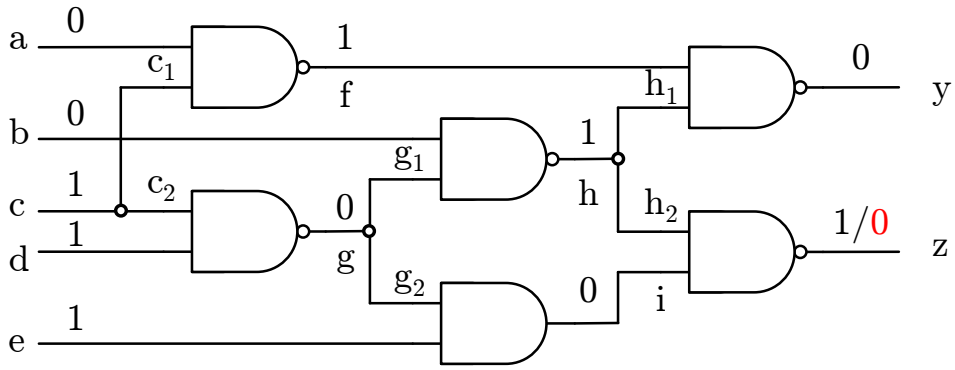


Figure 2.1: Example to illustrate back-coning and path tracing.

each signal is also shown in the figure. Assume that the output z is a failing output. The value at z is represented as v/v', where its correct (defective) value is v (v'). Back-coning from output z includes the signals $\{z, h_2, i, h, g_1, g_2, g, c_2, b, c, d, e\}$. Path tracing from output z includes the signals $\{z, i, g_2, g, c_2, c, d\}$.

Contrary to back-coning, potential defect locations implicated by path tracing can be different for each failing pattern. In the path tracing procedure [108–110], all signals encountered during tracing each failing pattern and from each failing output are probable defect locations. Path tracing typically finds many candidates, but the identified candidates overall represent a small fraction of all signals in the design.

An effect-cause technique scales well with design size and is guaranteed to include each defect location that assists in error propagation to at least one design output, even for multiple defects. However, such an analysis does not direct its efforts in identifying the behavior of a defect. In addition, because of its conservative nature, the diagnostic resolution for an effect-cause method is poor (i.e., it reports a large number of signals).

Various algorithms have also been put forward that benefit from using a combination of effect-cause and cause-effect analyses [108, 115–145]. That is, fault simulation is performed on the logical signals that are identified from an effect-cause method. One group of methods

use the notion of a composite signature [108, 115, 124–130].

The motive of a composite signature based approach is to create a signature that subsumes the possible behavior of a defect. A composite signature is formed by combining the signatures of more than one fault (typically, a stuck-at fault). For example, a composite response of a bridge fault between signals A and B is created by taking a union of the responses of the four stuck-at faults associated with the bridged signals [124–127]. A composite response of an open fault is constructed by taking a union of the responses of the stuck-at faults at the stem and the fan-out branches of a signal [108, 115]. A composite response can be generated from more than one candidate fault type as well [115]. A composite fault model typically generates a large set of candidates because there is no criterion to eliminate a candidate based on the number of mismatches between the response of the candidate and observed response.

Another group of methods that utilize the combination of effect-cause and cause-effect analyses perform per-test diagnosis [115,131–144]. Such techniques are collectively referred to as STAT (Single-Test-At-a-Time) techniques in [141]. Each failing pattern is independently used to generate candidates. A minimal set of candidates is then derived using some criterion. For example, in [136, 137], for each failing pattern, an effect-cause analysis (path tracing, for example) first identifies a region in the design where a defect could reside. Stuck-at faults in the identified region are then examined to determine the faults that explain that failing pattern. Such patterns are classified as SLAT (Single-Location-At-a-Time) patterns in [136,137]. A minimal set of stuck-at faults that collectively explains all the SLAT patterns is then derived.

The fundamental problem with logic-based diagnosis approaches is that they cannot pinpoint a physical location of a defect. To improve defect localization, diagnosis needs to use additional information other than the logic design itself. Design layout, which contains the coordinate information of every standard cell, pin and net, can be incorporated in the

diagnosis flow to enhance the quality of diagnosis. Hence, in addition to the information provided by logic diagnosis, physically-aware diagnosis [142–161] can further localize a defect by identifying potential defect locations and/or by improving the diagnostic resolution (both logical and physical [279]). For example, a pair of nets identified as a bridge defect by logic-based diagnosis is likely inaccurate if the two nets are never in close proximity [125, 142–144, 161, 278, 280]. In addition, a net identified as an open by logic-based diagnosis may span various metal layers while physically-aware diagnosis may improve physical resolution by eliminating some of the metal layers as a possible defect location [145, 147, 148, 150, 153–155, 158, 159, 161]. Back-end layout analysis techniques, which concentrate on further improving the physical resolution for a candidate by taking advantage of its neighborhood topology and/or various circuit parameters, are further scrutinized in Section 4.1.

Prior work discussed up to this point are based on binary error models, i.e., fault models that assume an erroneous value of 0 or 1 at a candidate location (such as the stuck-at fault model). However, not all defects (e.g., byzantine bridges [72] and opens [147]), even for a subset of failing patterns, necessarily behave like a stuck-at fault.

To resolve this shortcoming and avoid eliminating an actual defect location, the X-fault model [65] can be employed to represent the fault effect. X-fault simulation is similar to stuck-at simulation except that instead of modeling a faulty 0 or 1 at a defect location, an unknown value (X) is assumed. Fault effects modeled as unknown values conservatively propagate from a defect location through the circuit to the outputs. Outputs with an X are assumed to be potentially erroneous. The X-value simulation of a candidate is said to explain a failing pattern observed on the tester if the erroneous circuit outputs are subsumed by the set of simulated outputs that possess an X value. The X-fault model has been utilized to locate a defect in [65, 281–284].

Candidates obtained from diagnosis are then typically ranked by comparing the response of the candidate with the observed circuit response. For example, the STAT algorithm

presented in [141] uses evidence theory [285, 286] to rank candidates. In [108, 115], the candidates are ranked based on the amount of matches and mismatches between the observed and the candidate. The ranking is rather straightforward. Each candidate is represented by three metrics.

$$f(c) = \left(|EXP_p|, \sum_{p \in P} |TFSF_o^p|, \sum_{p \in TFSF_p} |TPSF_o^p| \right) \tag{2.1}$$

where, EXP_p denotes the set of failing patterns explained by a candidate c, P denotes the total number of test patterns, $TFSF_o^p$ represents the set of Tester-Fail-Simulation-Fail (TFSF) outputs for pattern p, $TPSF_o^p$ represents the set of Tester-Pass-Simulation-Fail (TPSF) outputs for pattern p, and $TFSF_p$ denotes the set of TFSF patterns.

The set of candidates are then sorted by decreasing $|EXP_p|$ and $\sum_{p \in P} |TFSF_o^p|$, and increasing $\sum_{p \in TFSF_p} |TPSF_o^p|$ (in that order).

On the other hand, the work of [138,139] represents each candidate as a set of four metrics as follows.

$$f(c) = \left(\sum_{p \in P} |TFSF_o^p|, \sum_{p \in P} |TPSF_o^p|, \sum_{p \in P} |TFSP_o^p|, \sum_{p \in P} min(|TFSF_o^p|, |TPSF_o^p|) \right) \tag{2.2}$$

where, $TFSP_o^p$ represents the set of Tester-Fail-Simulation-Pass (TFSP) outputs for pattern p.

The set of candidates are then sorted by increasing α, where α is the fourth element of the tuple shown in Equation 2.2, decreasing $\sum_{p \in P} |TFSF_o^p|$, and increasing $\sum_{p \in P} |TPSF_o^p|$ and $\sum_{p \in P} |TFSP_o^p|$.

Commercial diagnosis tools employ a scoring formula as well. While the diagnosis tools offered by Cadence [287] and Synopsys [288] calculate the score of each candidate using

Equation 2.3, the diagnosis software offered by Mentor Graphics [289] computes the score for each candidate using Equation 2.4.

$$f(c) = \frac{\sum_{p \in P} |TFSF_o^p|}{\sum_{p \in P} |TFSF_o^p| + \beta \sum_{p \in P} |TPSF_o^p| + \sum_{p \in P} |TFSP_o^p|} \; (0 < \beta \leq 1) \tag{2.3}$$

$$f(c) = 70\frac{F}{F+P} + 10\frac{F}{F+100P} + 10\frac{F}{F+1000P} + 10F \tag{2.4}$$

where,

$$F = \frac{|EXP_p|}{\max_i |EXP_p(i)|} \frac{|\bigcup_i EXP_p(i)|}{|FP|}$$

$$P = \frac{|TPSF_p|}{|PP|}$$

where, $\max_i |EXP_p(i)|$ is the maximum number of failing patterns explained by any candidate, $\cup_i EXP_p(i)$ is the set of failing patterns explained by all the candidates, FP is the set of failing patterns in the applied test set, and PP is the set of passing patterns in the applied test set.

The scoring methods described in [108, 115–117, 120, 123, 125–127, 129, 134, 135, 138, 139, 141–144, 290–292], including the expressions shown in Equations 2.1 through 2.4 are based on a binary error fault model (i.e., where the error is assumed either 1 or 0). However, as reasoned earlier, the use of such a fault model might yield an incorrect set of candidates to begin with. Moreover, although the methods discussed in [65, 281–284] are based on the X-fault model, but due to its inherent flexibility and generality in deriving candidates, accuracy is achieved at the cost of resolution in those methods. More importantly, prior scoring methods use various candidate-ranking heuristics to identify the best candidates. They are independent of the type of the defect, and hence cannot capture the characteristics

specific to each defect type. Additionally, prior ranking expressions are created by intuition and domain knowledge, and hence are not guaranteed to work for every defect type, design and/or process node.

Conversely, instead of using an ad-hoc scoring technique, a data-driven scoring model can discover (or learn) latent correlations between the correct candidate and the tester response, and possibly achieve better diagnosis accuracy and resolution. The scoring model can be built separately for each defect type to capture its unique characteristics.

It is thus not surprising that machine learning (ML) has been applied in the area of test, diagnosis and yield analysis. Specifically, in the area of diagnosis, prior work can be divided into three categories. The first category exploits machine learning before diagnosis begins, to make diagnosis more efficient [293–298]. For instance, the works of [294, 295] utilize a random forest to assess the effectiveness of diagnosis *a priori* so that diagnosis resources can be allocated smartly. In [296, 297], different ML algorithms such as linear regression, the k-nearest neighbors algorithm, a support vector machine and a decision tree are used to predict the precise amount of test data necessary to obtain an accurate diagnosis. The work of [298] identifies a systematic defect from the test data by using hierarchical clustering.

The second category of methods applies machine learning during diagnosis, to make diagnosis more effective. For instance, a multi-class support vector machine is employed in [299, 300] to locate a defect in a failing chip, where each class represents a fan-out free region in the design. A large design can be divided into smaller partitions to reduce memory and runtime overhead. In [301], a random forest is used to predict whether a failing chip is affected by a bridge defect. In that work, several features for each candidate are extracted from the design and the test data. The features identified from a logic description of the design aim to differentiate a bridge from other defect types. For example, one feature checks if the bridged nets drive a parity cell; a short between the inputs of a parity cell is equivalent to a stuck-at fault at its output, and thus adds ambiguity to bridge defect

prediction. The features extracted from the test data indicate how well the simulation response of a bridge candidate matches with the observed tester response. The approach proposed in [302] accomplishes a similar objective of defect classification by training a neural network. However, the methods presented in [301, 302] do not identify the defect location, which means they do not improve candidate-level resolution. Machine learning has been effective in diagnosing scan chain failures as well [303, 304].

The third category of methods applies machine learning after diagnosis to aid subsequent failure analysis and possibly accelerate yield learning [158,159,221–232,242,243,305–308]. For example, a support vector machine is used in [306–308] on volume diagnosis data to improve the resolution of each individual diagnosis. In addition to the typical features identified from the test data such as the number of TFSF, TPSF and TFSP patterns and outputs, the novel component of that work is the inclusion of physical features associated with defect excitation such as the number of failing and passing neighborhood states for a candidate. The work of [231, 232] attempts to find failure-causing layout geometries by clustering layout regions suspected by diagnosis, which, in turn, can aid in identifying a potential systematic defect. An expectation-maximization algorithm is employed in [242,243] to pinpoint the Design-for-Manufacturing (DFM) rule whose violation causes a chip to fail, and in [225, 230, 305] to find the failure root-cause distribution of the failing population. The latter has also been estimated using statistical hypothesis testing in [159, 227] and a supervised learning method in [226].

Because LearnX applies machine learning during diagnosis, each aforementioned method that is executed before diagnosis and after diagnosis complements LearnX. Except the work presented in [299, 300], where resolution is limited to a fanout-free region and hence inadequate for subsequent failure analysis, prior methods that apply machine learning during diagnosis do not improve resolution at a candidate-level, and hence strengthen LearnX. For instance, the methods introduced in [301,302] predict the type of a defect, but not the defect

candidate itself. Additionally, the approach proposed in [303, 304] target scan chain defects, whereas LearnX attempts to locate a defect that resides in a logic circuit. Thus, those methods enhance LearnX as well.

To summarize, numerous diagnosis algorithms have been put forward over the years to improve defect localization. Cause-effect based approaches build a database of simulated faulty behaviors and compare them with the observed tester response. Effect-cause based approaches deduce one or more defects by examining the observed circuit behavior. Modern approaches incline towards employing a combination of these analyses to improve diagnosis quality. Such an approach either uses a composite fault model or performs per-test diagnosis. However, a defect that causes multiple faulty signals can escape diagnosis based on a binary error model. Instead, the X-fault model can be used to represent the fault effect to avoid discarding an actual defect location. Additionally, ad-hoc scoring heuristics are designed in prior work to rank candidates to improve logical resolution, and are thus not assured to work consistently and effectively. On the other hand, machine learning has been successfully exploited in various areas of test, diagnosis and yield learning, and offers a viable alternative for candidate prediction.

2.2 Diagnosis Methodology

An enhanced diagnosis procedure is extremely important for improving design, test and manufacturing of a chip. It makes volume diagnosis, and subsequently, PFA more effective in their ability to pinpoint and verify the most probable cause of yield loss. LearnX is a step in that direction.

To address the shortcomings associated with different aspects of diagnosis discussed in Section 2.1, this section describes a single-chip diagnosis methodology that we term LearnX. Salient features of LearnX include:

1. It is a physically-aware diagnosis approach that utilizes layout information to identify not only the defect location but also its physical defect type (e.g., interconnect open, interconnect bridge and cell-internal defect). It should be noted that LearnX complements/strengthens approaches like [146, 162] where physical resolution is improved using layout neighborhood analysis.

2. Instead of just using a deterministic fault model where an erroneous value of either 0 or 1 is assumed, it employs the X-fault model as well because it is immune to error masking. As a result, it allows an error to propagate from a defect location conservatively, which likely avoids removing a correct candidate.

3. It applies machine learning to generate a scoring model to uncover the hidden correlations between a candidate and the tester response for identifying the candidate that best represents a defect. Specifically, a supervised learning algorithm is used to distinguish the correct candidate from incorrect ones.

4. It is applicable to defects exhibiting arbitrary misbehaviors.

Figure 2.2 shows the overview of LearnX. LearnX is a two-phase diagnosis methodology. The first phase (detailed in Section 2.2.1) aims to identify a defect that mimics the behavior of classic fault models such as the stuck-at, the bridge (specifically, the AND-type, the OR-type and the dominating bridge) and the open fault model (where a net is assumed to be stuck at the opposite value of the expected value for each pattern) through a set of strict rules. These strict rules must ensure that (a) the actual defect behavior and location are accurately captured by one of the identified candidates and (b) the minimum number of candidates are reported.

The defects that do not satisfy these rules are diagnosed using the steps outlined in the second phase of the methodology (detailed in Section 2.2.3). Such defects are identified

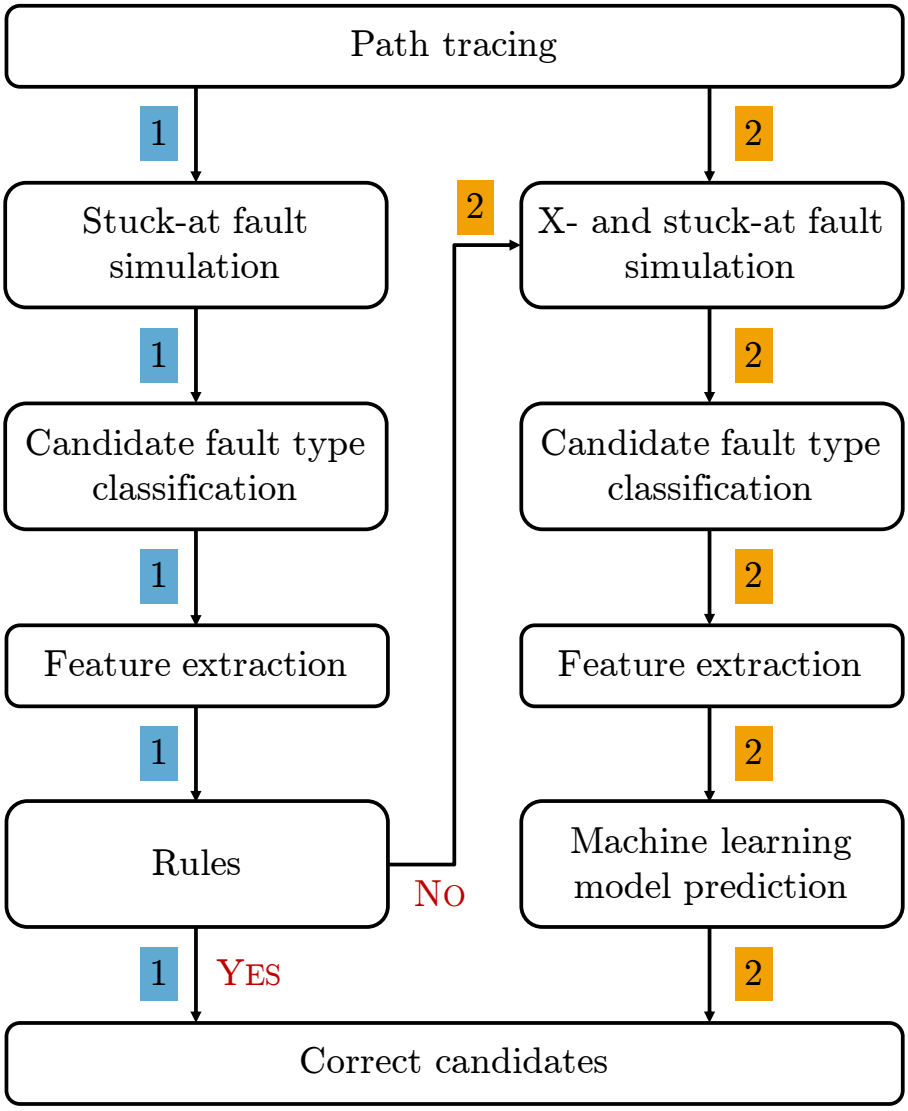


Figure 2.2: An overview of the proposed diagnosis methodology, LearnX. The sequence of steps in each phase are marked with a different color.

to have a non-trivial behavior and cannot be modeled using the fault models employed in Phase 1. The two main steps in Phase 2 are (a) fault simulation using the X-fault model [281], which aims to capture the complex behavior of a defect with relatively high accuracy, and (b) machine learning classification to distinguish between the correct and the incorrect candidates. Section 2.2.2 overviews the application of machine learning, and

specifically, supervised learning, in the context of candidate classification.

2.2.1 Phase 1

The first step in Phase 1 of LearnX is path tracing [108–110]. For each failing pattern, path tracing starts from each erroneous circuit output and traces back through the circuit towards the inputs, deducing the potential defective (logical) signals along the way. Physical defect locations corresponding to each implicated logical location are then extracted from layout analysis. Specifically, the topology and the physical neighborhood of each net are examined to identify probable open and bridge defect locations [146].

Next, for each failing pattern, a stuck-at fault at each candidate location is simulated to find faults that explain that pattern. Sets of stuck-at faults are selected using the minimum set-cover algorithm such that faults in each cover explain all the failing patterns.

Each candidate cover is classified into one of the following four fault types according to its behavior.

1. STUCK: When the cover contains only one (stuck-at) fault, the cover is classified as STUCK. Here, the candidate location is also checked against the pre-extracted set of likely defect locations to examine if an open or a bridge defect is feasible at this location.

2. CELL: If the cover consists of fault(s) (of either one or both polarities) at the output of a standard cell, the cell driving the fault location is analyzed further to determine whether a cell defect is possible. This analysis, referred to as the cell consistency check, is based on the realistic assumption that excitation of a cell-internal defect is highly correlated to the input logic values applied to a cell [132,142–144,171]. Cell consistency check is discussed in detail in Section 4.1.2.

3. BRIDGE: When the cover comprises of faults corresponding to two physically adjacent nets, the candidate cover is classified as BRIDGE.

4. OPEN: If the cover consists of more than one fault affecting the same signal, i.e., if the cover comprises of faults at the stem and/or one or more of its fan-out branches, the cover is classified as OPEN. The cover is also validated against the physical feasibility of being an open defect location. Additionally, physical neighbors of the net are monitored to check the existence of a dominant driver.

Next, a set of features for each candidate cover is extracted according to its fault type. Phase 1 particularly deals with defects that strongly reflect the characteristics of a stuck-at, a cell, a wired-OR, a wired-AND, a dominant bridge fault, or an open fault (where a net is assumed to be stuck at the opposite value of the expected value for each pattern). The extracted features considered are summarized in Table 2.1. It should be noted that for candidate covers of size greater than one, feature derivation is based on the stuck-at faults that constitute the cover.

Based on the features shown in Table 2.1, rules are constructed for each fault type to identify a defect that mimics the behavior of well-known fault models. The designed rules are shown in Table 2.2. These rules are chosen based on the test and manufacturing domain knowledge. Specifically, for fault type,

1. STUCK: The rules imply that the fault simulation response is identical to the observed response on the tester.

2. CELL: The first two rules imply that each failing pattern is explained by the cell fault. The third rule examines the cell for consistency. Specifically, if the sets of logic values established on the cell inputs for Tester-Fail-Simulation-Fail (TFSF) and Tester-Pass-Simulation-Fail (TPSF) are disjoint, then the candidate is deemed consistent.

Fault type	Feature	Feature description
STUCK	$\sum_{p\in P} \lvert TFSF_o^p \rvert$	Number of TFSF outputs
	$\sum_{p\in P} \lvert TPSF_o^p \rvert$	Number of TPSF outputs
	$\sum_{p\in P} \lvert TFSP_o^p \rvert$	Number of TFSP outputs
CELL	$\sum_{p\in P} \lvert TFSF_o^p \rvert$	Number of TFSF outputs
	$\sum_{p\in P} \lvert TFSP_o^p \rvert$	Number of TFSP outputs
	$\bigcup_{p\in TFSF_p} S_{cell}^p$	Set of failing states for the cell inputs
	$\bigcup_{p\in TPSF_p} S_{cell}^p$	Set of passing states for the cell inputs
BRIDGE	$\sum_{p\in P} \lvert TFSF_o^p \rvert$	Number of TFSF outputs
	$\sum_{p\in P} \lvert TFSP_o^p \rvert$	Number of TFSP outputs
	$\bigcup_{p\in TFSF_p} S_{bridge}^p$	Set of failing states for the bridged nets
	$\bigcup_{p\in TPSF_p} S_{bridge}^p$	Set of passing states for the bridged nets
OPEN	$\sum_{p\in P} \lvert TFSF_o^p \rvert$	Number of TFSF outputs
	$\sum_{p\in P} \lvert TPSF_o^p \rvert$	Number of TPSF outputs
	$\sum_{p\in P} \lvert TFSP_o^p \rvert$	Number of TFSP outputs

P is the number of patterns in the applied test set.
$TFSF_o^p$ is the set of TFSF outputs for a pattern, p.
$TPSF_o^p$ is the set of TPSF outputs for a pattern, p.
$TFSP_o^p$ is the set of TFSP outputs for a pattern, p.
S_{cell}^p is the set of logic values established on the cell inputs for a pattern, p.
S_{bridge}^p is the set of logic values established on the bridged nets for a pattern, p.

Table 2.1: Features extracted from the test data for Phase 1.

3. BRIDGE: The first two rules imply that each failing pattern is explained by the faults at the bridge location. The third rule ensures that the bridged nets have opposite polarities for TFSF patterns, while the fourth rule confirms if the nets have same polarities (i.e., bridge is not excited) for TPSF patterns.

4. OPEN: The rules imply that the fault simulation response and the observed circuit response are the same.

Fault type	Rules
STUCK	$\sum_{p\in P} \lvert TFSF_o^p \rvert = \sum_{p\in P} \lvert TF_o^p \rvert$ $\sum_{p\in P} \lvert TPSF_o^p \rvert = 0$ $\sum_{p\in P} \lvert TFSP_o^p \rvert = 0$
CELL	$\sum_{p\in P} \lvert TFSF_o^p \rvert = \sum_{p\in P} \lvert TF_o^p \rvert$ $\sum_{p\in P} \lvert TFSP_o^p \rvert = 0$ $\bigcup_{p\in TFSF_p} S_{cell}^p \bigcap \bigcup_{p\in TPSF_p} S_{cell}^p = \emptyset$
BRIDGE	$\sum_{p\in P} \lvert TFSF_o^p \rvert = \sum\limits_{p\in P} \lvert TF_o^p \rvert$ $\sum_{p\in P} \lvert TFSP_o^p \rvert = 0$ $\bigcup_{p\in TFSF_p} S_{bridge}^p \in \{\{01\},\{10\},\{01,10\}\}$ $\bigcup_{p\in TPSF_p} S_{bridge}^p \in \{\emptyset,\{00\},\{11\},\{00,11\}\}$
OPEN	$\sum_{p\in P} \lvert TFSF_o^p \rvert = \sum_{p\in P} \lvert TF_o^p \rvert$ $\sum_{p\in P} \lvert TPSF_o^p \rvert = 0$ $\sum_{p\in P} \lvert TFSP_o^p \rvert = 0$

Table 2.2: Rule criteria for Phase 1.

A candidate that satisfies the rules tabulated in Table 2.2 is deemed correct. If no candidate of a failing chip complies with these rules, it is passed on to the second phase of the diagnosis flow that especially deals with defects having complex behavior.

Before presenting Phase 2, Section 2.2.2 discusses how machine learning, and specifically, supervised learning, can be leveraged for candidate classification.

2.2.2 Supervised Machine Learning

Figure 2.3 illustrates a high-level overview of how machine learning (supervised machine learning, in particular) is applied here to accomplish candidate prediction.

The first step for building a machine learning model in this context is to generate virtual fail logs. Numerous fail logs are created by injecting and simulating realistic defect

behaviors. Bridge, open and cell[2] defects with known and arbitrary behaviors are considered when creating virtual fail logs. Each fail log is then diagnosed by LearnX (up to the step named "feature extraction" in Figure 2.2) to obtain an initial set of candidates. Features are extracted for each candidate by comparing its simulation response with the observed tester response (i.e., the simulation response of the injected defect). Because the location of an injected defect is known *a priori*, each candidate is then labeled either as correct or incorrect. The resulting dataset consisting of the features and label (i.e., whether a candidate is correct or not) for each candidate is referred to as the training dataset.

A supervised learning algorithm examines the training dataset to build a prediction model. Given an undiagnosed fail log, a set of candidates is obtained by following the flow of Figure 2.2 up to the step named "feature extraction". The generated prediction model classifies each candidate as correct/incorrect. A commonly used machine learning method called a random forest [309] is utilized in this work. A random forest can be used for classification as well as regression. Because our objective is to find whether a candidate represents an actual defect or not, we use a random forest as a classifier.

A random forest is an ensemble learning method that constructs a "forest" of decision trees during training. A decision tree is built by labeling its each node with a decision or a condition that is used to partition the tree into smaller data sets. The process of splitting the tree continues until the data subset only contains the samples of the same class (or when some other stopping criterion is met). Decision trees are straightforward and easy to interpret, but face a few limitations. A decision tree is not robust; a small change in the training data can lead to a large number of mispredictions. Moreover, it is prone to overfitting, i.e., it does not generalize well from the training data. A random forest addresses these shortcomings. A random forest is a combination of multiple trees, where each tree is trained

[2]Realistic cell-internal defects are simulated at a transistor-level to obtain a cell-level defect response. A fail log is then produced by simulating the design with the *modified* cell function.

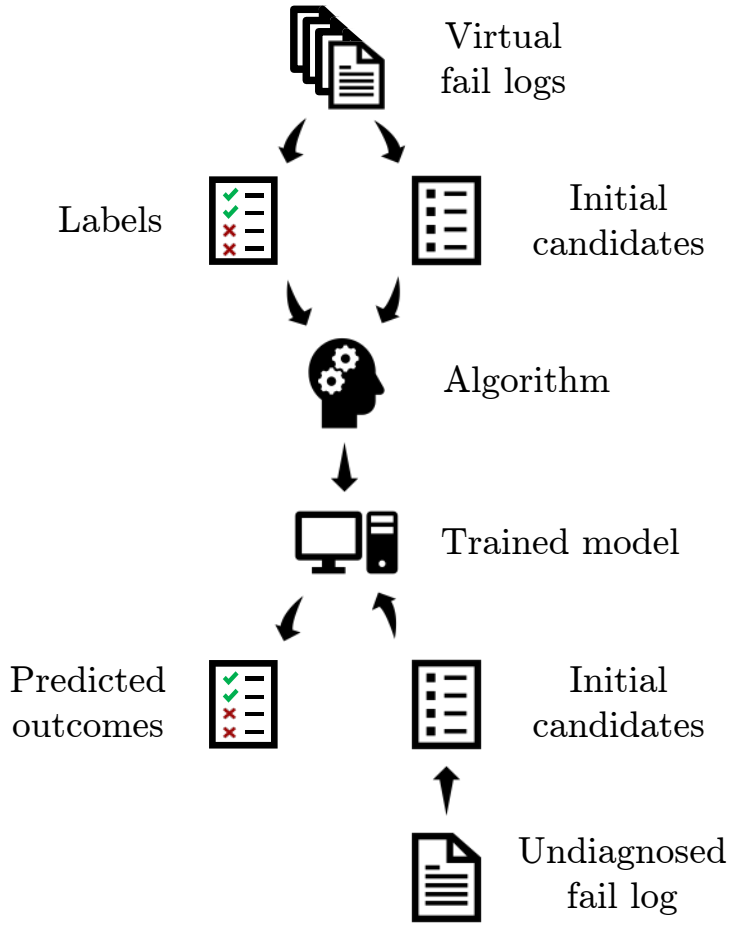


Figure 2.3: An overview of how supervised machine learning is applied in LearnX.

with a different subset of training samples and a different subset of features. Classification is (usually) performed by taking a majority vote over a set of classifications produced by the forest of trees.

A learning algorithm, including a random forest, has certain parameters (called hyperparameters) that can be adjusted to optimize its prediction effectiveness. The hyperparameters of a random forest include the number of trees, the depth of each tree and the random feature selected at each node for partitioning the tree. A common approach for hyperparameter tuning is cross validation.

Note that the training dataset is highly imbalanced. For each injected fault, there is only one correct candidate. Therefore, it is entirely possible that the default classification/decision threshold of 0.5 (which corresponds to majority voting for a random forest algorithm) is not optimum. The analysis of a receiver operating characteristic (ROC) curve and a precision-recall (PR) curve are two common techniques to derive the optimum threshold. A ROC curve illustrates the variation of the true positive rate[3] (also called the recall) with the false positive rate[4] for various thresholds. A PR curve, on the other hand, shows the trade-off between precision, which is the proportion of the instances that are truly positive among the ones classified as positive, and recall, for various thresholds. A PR curve is preferred over a ROC curve when the total number of positive instances is far fewer than the total number of negative instances [310]. Therefore, a PR curve is used here to find the optimum decision threshold for a random forest.

Figure 2.4 illustrates the effect of increasing threshold on precision and recall. In this work, the optimum threshold for each trained forest is found by maximizing the F1-score [310] (plotted with a black dotted line in Figure 2.4), which is defined as the harmonic mean of precision and recall.

2.2.3 Phase 2

Phase 2 begins similarly to Phase 1 with path tracing. Next, each candidate is simulated for each failing pattern using the X-fault model [65]. The resulting simulation responses are compared to the tester response to find candidates that explain that pattern. The X-value simulation of a candidate is said to explain a failing pattern if the erroneous circuit outputs are subsumed by the set of simulated outputs that possess an X value. Then, sets of X faults are selected using the minimum set-cover algorithm such that faults in each cover

[3]True positive rate, or recall, is defined as the ratio of positive instances that are correctly classified.
[4]False positive rate is defined as the ratio of negative instances that are incorrectly classified.

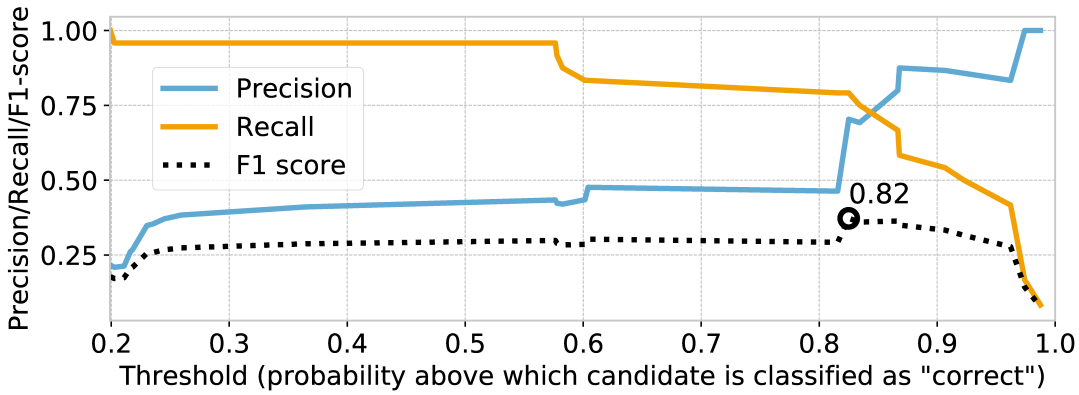


Figure 2.4: Selecting the optimum decision threshold that maximizes the harmonic average between precision and recall (known as the F1-score).

collectively explain all the failing patterns.

Each candidate cover is further analyzed using stuck-at simulation. Specifically, for each failing pattern, stuck-at faults at the locations corresponding to the X faults in a cover are simulated. The stuck-at fault responses are then compared to the observed circuit response to find the fault that best explains that pattern. Here, the criterion for best explaining a pattern is that the hamming distance between the fault simulation response and observed test response is minimum. Thus, each candidate, up to this point, is characterized by a cover of X faults and a cover of stuck-at faults.

The next step in Phase 2 of LearnX is assigning a fault type (STUCK, CELL, BRIDGE, and OPEN) to each candidate, which is accomplished in exactly the same way as Phase 1, and is described in Section 2.2.1.

Next, a set of features for each candidate is extracted using test and manufacturing domain knowledge. The pattern-level features are shown in Table 2.3, and the output-level features are shown in Table 2.4. The extracted set of features are specific properties of a candidate that aim to distinguish a correct candidate from an incorrect one. Each feature value is calculated by comparing the test outputs/patterns observed on the tester

Feature	Feature description
$\|TFSF_p\|/\|TF_p\|$	Ratio of the number of TFSF patterns to the number of TF patterns
$\|TFSF_p\|/\|SF_p\|$	Ratio of the number of TFSF patterns to the number of SF patterns
$\|TPSF_p\|/\|TP_p\|$	Ratio of the number of TPSF patterns to the number of TP patterns
$\|TPSF_p\|/\|SF_p\|$	Ratio of the number of TPSF patterns to the number of SF patterns
$\|TFSP_p\|/\|TF_p\|$	Ratio of the number of TFSP patterns to the number of TF patterns
$\|TFSP_p\|/\|SP_p\|$	Ratio of the number of TFSP patterns to the number of SP patterns

Table 2.3: Pattern-level features extracted from the test data for Phase 2. Six features for each type of simulation are identified, resulting in a total of 12 pattern-level features.

and predicted by simulation. Unlike other scoring methods in the literature, the features used here are derived from both the X-fault and the stuck-at fault simulation of a candidate[5], and are thus believed to capture a more complete picture. In addition, the features extracted here are more detailed. For example, instead of recording the total number of TPSF outputs over patterns that fail during simulation, the number of TPSF outputs are counted separately for TFSF and TPSF patterns (rows 3-4 and 9-10 in Table 2.4, respectively). Similarly, the number of TFSP outputs are noted separately for TFSF and TFSP patterns as well.

The next step in Phase 2 is to classify whether a candidate is correct or not using machine learning. As described in Section 2.2.2, a commonly used machine learning algorithm called a random forest [309] is utilized for this purpose. Specifically, four different random forests are generated – one for each fault type, with the objective of learning specific attributes pertaining to each fault behavior. Figure 2.5 illustrates the overall flow for training four

[5]Twenty-two features listed in Tables 2.3 and 2.4 are extracted for each type of simulation, resulting in a total of 44 features.

Feature	Feature description
$\sum_{p\in TFSF_p} \lvert TFSF_o^p\rvert / \sum_{p\in P} \lvert TF_o^p\rvert$	Ratio of the number of TFSF outputs for TFSF patterns to the number of TF outputs
$\sum_{p\in TFSF_p} \lvert TFSF_o^p\rvert / \sum_{p\in P} \lvert SF_o^p\rvert$	Ratio of the number of TFSF outputs for TFSF patterns to the number of SF outputs
$\sum_{p\in TFSF_p} \lvert TPSF_o^p\rvert / \sum_{p\in P} \lvert TP_o^p\rvert$	Ratio of the number of TPSF outputs for TFSF patterns to the number of TP outputs
$\sum_{p\in TFSF_p} \lvert TPSF_o^p\rvert / \sum_{p\in P} \lvert SF_o^p\rvert$	Ratio of the number of TPSF outputs for TFSF patterns to the number of SF outputs
$\sum_{p\in TFSF_p} \lvert TFSP_o^p\rvert / \sum_{p\in P} \lvert TF_o^p\rvert$	Ratio of the number of TFSP outputs for TFSF patterns to the number of TF outputs
$\sum_{p\in TFSF_p} \lvert TFSP_o^p\rvert / \sum_{p\in P} \lvert SP_o^p\rvert$	Ratio of the number of TFSP outputs for TFSF patterns to the number of SP outputs
$\sum_{p\in TFSF_p} \lvert TPSP_o^p\rvert / \sum_{p\in P} \lvert TP_o^p\rvert$	Ratio of the number of TPSP outputs for TFSF patterns to the number of TP outputs
$\sum_{p\in TFSF_p} \lvert TPSP_o^p\rvert / \sum_{p\in P} \lvert SP_o^p\rvert$	Ratio of the number of TPSP outputs for TFSF patterns to the number of SP outputs
$\sum_{p\in TPSF_p} \lvert TPSF_o^p\rvert / \sum_{p\in P} \lvert TP_o^p\rvert$	Ratio of the number of TPSF outputs for TPSF patterns to the number of TP outputs
$\sum_{p\in TPSF_p} \lvert TPSF_o^p\rvert / \sum_{p\in P} \lvert SF_o^p\rvert$	Ratio of the number of TPSF outputs for TPSF patterns to the number of SF outputs
$\sum_{p\in TPSF_p} \lvert TPSP_o^p\rvert / \sum_{p\in P} \lvert TP_o^p\rvert$	Ratio of the number of TPSP outputs for TPSF patterns to the number of TP outputs
$\sum_{p\in TPSF_p} \lvert TPSP_o^p\rvert / \sum_{p\in P} \lvert SP_o^p\rvert$	Ratio of the number of TPSP outputs for TPSF patterns to the number of SP outputs
$\sum_{p\in TFSP_p} \lvert TFSP_o^p\rvert / \sum_{p\in P} \lvert TF_o^p\rvert$	Ratio of the number of TFSP outputs for TFSP patterns to the number of TF outputs
$\sum_{p\in TFSP_p} \lvert TFSP_o^p\rvert / \sum_{p\in P} \lvert SP_o^p\rvert$	Ratio of the number of TFSP outputs for TFSP patterns to the number of SP outputs
$\sum_{p\in TFSP_p} \lvert TPSP_o^p\rvert / \sum_{p\in P} \lvert TP_o^p\rvert$	Ratio of the number of TPSP outputs for TFSP patterns to the number of TP outputs
$\sum_{p\in TFSP_p} \lvert TPSP_o^p\rvert / \sum_{p\in P} \lvert SP_o^p\rvert$	Ratio of the number of TPSP outputs for TFSP patterns to the number of SP outputs

Table 2.4: Output-level features extracted from the test data for Phase 2. Sixteen features for each type of simulation are identified, resulting in a total of 32 output-level features.

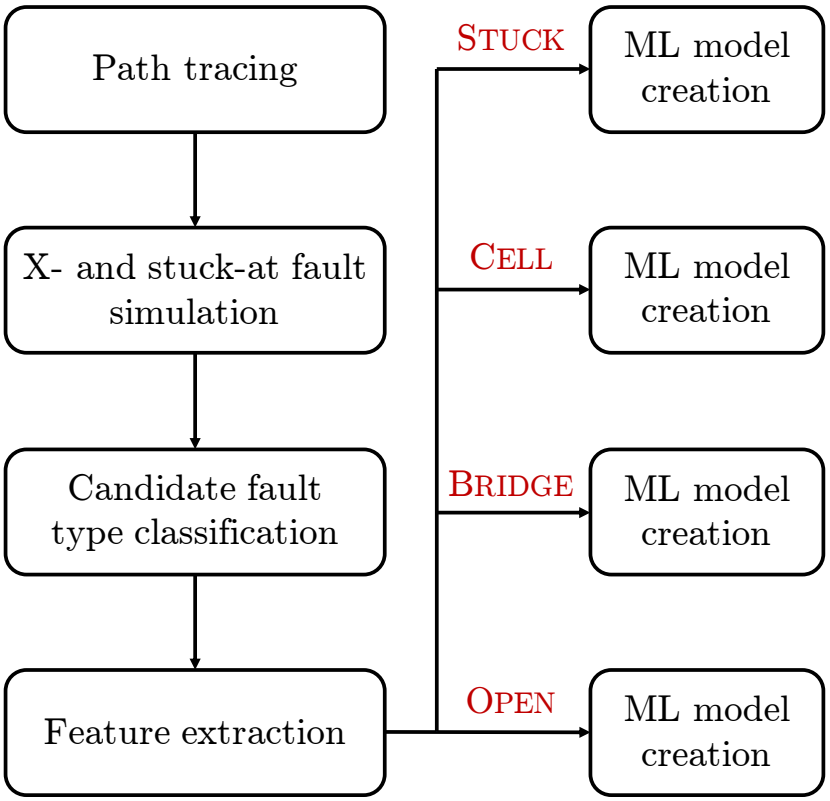


Figure 2.5: Overview of the flow employed for creating a machine learning model for each fault type.

machine learning models, one for each fault type (specifically, STUCK, CELL, BRIDGE and OPEN).

Each random forest is trained using virtual test responses generated through fault injection and simulation. Hyperparameters of each trained model are tuned using cross validation. Because only one correct candidate exists for each virtual fail log, the training data is highly imbalanced. Therefore, an optimum decision threshold for each random forest is derived using a PR curve. The process of creating a learning model is explained in Section 2.2.2.

Each learned model inherently acts like a scoring framework, and assigns a probability (or a "score") to each defect candidate. The score assigned to each candidate represents the likelihood of a candidate being correct. Any candidate whose score is more than the decision

threshold is deemed a correct candidate.

To summarize, LearnX is a two-phase diagnosis methodology that characterizes a defect with respect to its physical location and behavior. The first phase diagnoses a defect through a set of rules that are aimed towards identifying a defect that mimics known fault behaviors. Undiagnosed defects are then analyzed using the second phase, where machine learning (along with the X-fault model that propagates error conservatively to avoid eliminating a correct candidate) is applied to learn characteristics that differentiates correct candidates from incorrect ones.

2.3 Experiments

This section describes two experiments to validate LearnX: one is a defect injection and simulation experiment using four different designs (Section 2.3.2) and another that uses real silicon failure data (Section 2.3.3).

The diagnostic metrics that are used to quantify the effectiveness of LearnX are discussed in Section 2.3.1. In each experiment, the flow of Figure 2.2 is used to analyze a fail log, where the steps illustrated in Figure 2.5 are used to create a learning model for each fault type (namely, Stuck, Cell, Bridge and Open). To gauge the performance of LearnX, each fail log is also analyzed with state-of-the-art commercial diagnosis, where only the top-scoring candidates are considered at its output.

2.3.1 Diagnostic Metrics

A diagnosis methodology should ideally (a) report a single candidate that correctly represents the defect residing in a failing chip, and (b) identify the type and characterize the logic behavior of the reported candidate. It should be recalled that LearnX is the first stage of our diagnosis methodology and concentrates on defect localization. LearnX reports a

candidate at the logic level[6]. It filters out any candidate that is physically infeasible (e.g., a bridge defect between two nets that are physically distant from each other is unlikely). It can also differentiate between a dominant bridge and an open defect candidate by tracking the logic values of the neighbors of the "victim" net.

To that end, the following metrics are employed to evaluate the effectiveness of LearnX.

- Accuracy: A defect is said to be accurately diagnosed if its location matches with the location of one of the reported candidates. Specifically, a cell defect is deemed accurate if diagnosis reports the corresponding standard cell; an open defect is regarded accurate if the net associated with the defect is correctly suspected; and a bridge defect is considered accurate if both the bridged nets are correctly identified.

- Resolution: Resolution is defined as the number of defect candidates suspected by diagnosis[7].

- *Home run*: A diagnosis approach is said to hit a home run when an ideal diagnosis outcome is returned, i.e., when a single correct candidate is reported. Specifically, *home run* is equal to one when a defect is accurately identified and its resolution is one, and zero otherwise.

2.3.2 Simulation Experiment

For assessing LearnX, a simulation-based experiment is performed using four different designs. One is an Advanced Encryption Standard (AES) core that provides AES-128 encryption (henceforth referred to as "AES") [311], the second design is an IWLS'05 benchmark

[6]PADLOC and NOIDA, the remaining two stages of our diagnosis methodology, further analyze the output of LearnX. Specifically, PADLOC (NOIDA) focuses on localizing a back-end (front-end) defect at the physical level, that is, it implicates a region in the design layout for a defect candidate, and derives its precise logic behavior. PADLOC (NOIDA) is described in detail in Chapter 4 (Chapter 5).

[7]A candidate of type BRIDGE, which consists of two nets, is counted as a single candidate instead of two. Thus, the corresponding resolution is equal to one.

that implements the Data Encryption Standard (DES) algorithm called *des_perf* (henceforth referred to as "DES"), the third design is the L2 cache of the OpenSPARC T2 processor (called "L2B") [312], and the fourth design is an ITC'99 benchmark circuit called "B18"[8] [313]. Each circuit is synthesized, and placed and routed using a 45nm technology library [314].

For each design, the following fault types are considered to model realistic defect behaviors.

1. Two-line bridge defects: A variety of bridge fault models including the wired-bridge model (AND-type, OR-type and the dominating bridge) and the biased voting model [72] are adopted to model a bridge defect. In the biased voting model, the voltage of the bridged nets depends on the relative strengths of the standard cells driving the bridged nets and the surrounding circuitry. This voltage can be interpreted as either logic-1 or logic-0 based on the switching threshold of the receiver cell inputs. As a result, an error can manifest at one of the nets for some failing patterns, at the other bridged net for some other failing patterns, and even at both the nets simultaneously for some failing patterns. Potential bridge net pairs are extracted from the design layout via geometric proximity analysis [146].

2. Open defects: The voltage at the floating net is influenced by various factors such as the driving strength of the net, the threshold voltages of the receiver cell inputs, and logic values at the neighboring interconnects [145–155]. The voltage can be interpreted differently by each receiver standard cell. Here, to model this unpredictable behavior, an open defect is modeled in two ways; by creating a composite fault signature from the stuck-at faults of both the polarities (at each defect location), and by assuming that a (possibly different) subset of the fan-out branches are erroneous at an open

[8] B18 consists of two copies of a subset of the Viper processor and six copies of a subset of the Intel 80386 processor.

defect location for each sensitizing pattern [147]. Open defect locations are extracted from the layout by examining the topology of each net [146].

3. Cell defects: Open, bridge and transistor defects are injected into the layout of each standard cell. (Defect injection procedure inside a cell is further explained in Section 5.4.2.) Each resulting cell-level defect response is then simulated at the logic level to produce a "fail log".

Each defect behavior is modeled using fault tuples [315, 316]. For each design, a total of 7,000 virtual fail logs are created. A test set that achieves 100.0% stuck-at fault efficiency is generated using a commercial ATPG tool. Each faulty circuit is simulated at the logic level using FATSIM [317] to produce a "tester response".

For each design, out of 7,000 fail logs, 1,000 are used to create the training dataset. Thus, 6,000 fail logs are used for the test dataset. Each fail log is diagnosed using LearnX and two state-of-the-art commercial diagnosis tools. All three diagnosis techniques are evaluated on the criteria described in Section 2.3.1.

Table 2.5 shows the number of fail logs diagnosed by Phase 1 and Phase 2 for each design and for each injected defect type. It is clearly seen that each fail log associated with a cell defect is diagnosed by Phase 1. The percentage of defects diagnosed by Phase 2 varies from 37.5% for L2B to 45.4% for AES. Specifically, the percentage of bridge defects analyzed by Phase 2 varies from 56.3% for L2B to 64.0% for AES. The variation in the number of open defects examined by Phase 2 is more; while Phase 2 inspects 56.2% of open defects for DES and L2B, it analyzes 72.3% of open defects for AES. On average over all designs, 40.5% of defects are diagnosed by Phase 2. Among these defects, 60.4% are bridge defects and 61.3% are open defects.

Section 2.3.2.1 and Section 2.3.2.2 discuss the performance of Phase 1 and Phase 2, respectively. Section 2.3.2.3 summarizes the overall effectiveness of LearnX.

	Phase 1				Phase 2			
Design	Bridge	Open	Cell	Total	Bridge	Open	Cell	Total
AES	720	554	2000	3274	1280	1446	0	2726
DES	792	876	2000	3668	1208	1124	0	2332
L2B	875	877	2000	3752	1125	1123	0	2248
B18	783	792	2000	3575	1217	1208	0	2425

Table 2.5: Number of defects of each type diagnosed by Phase 1 and Phase 2 for four designs.

2.3.2.1 Phase 1

Figures 2.6 through 2.9 show the histograms of the resolution achieved by Phase 1 for the four designs examined. The x-axis shows the resolution and the y-axis shows the number of fail logs. The percentage of fail logs accurately diagnosed for a particular resolution is shown at the top of its corresponding plot-bar. The top half of each figure (i.e., above $y = 0$) shows the distribution of the number of fail logs that are correctly diagnosed while the bottom half shows the distribution of the inaccurately diagnosed fail logs. The percentage of fail logs diagnosed accurately by each diagnosis technique is shown above the plot in each figure.

Observations specific to each design are as follows.

1. **AES**: Figure 2.6 reveals that the average accuracy for Phase 1 is 98.2%, which is 0.9% (15.0%) more than Tool 1 (Tool 2). In addition, Phase 1 attains perfect resolution for 44.5% of fail logs, of which 97.8% are correct. On the other hand, Tool 1 (Tool 2) is 97.7% (81.7%) correct among the defects diagnosed with perfect resolution. Moreover, Phase 1 (and Tool 2) delivers a home run (i.e., returns an ideal diagnosis) for 43.5% of fail logs, which is 4.1% better than Tool 1.

2. **DES**: It is observed from Figure 2.7 that Phase 1 correctly locates a defect for 99.1% of fail logs, while Tool 1 (Tool 2) correctly locates a defect for 97.9% (81.3%) of fail

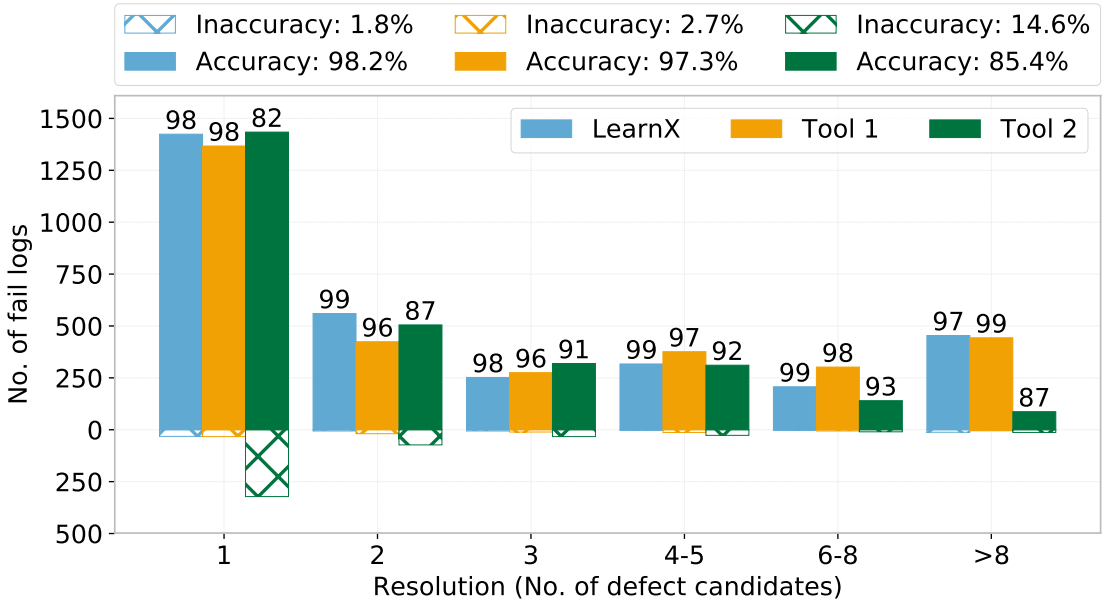


Figure 2.6: Resolution distribution attained by LearnX (Phase 1) for the design "AES".

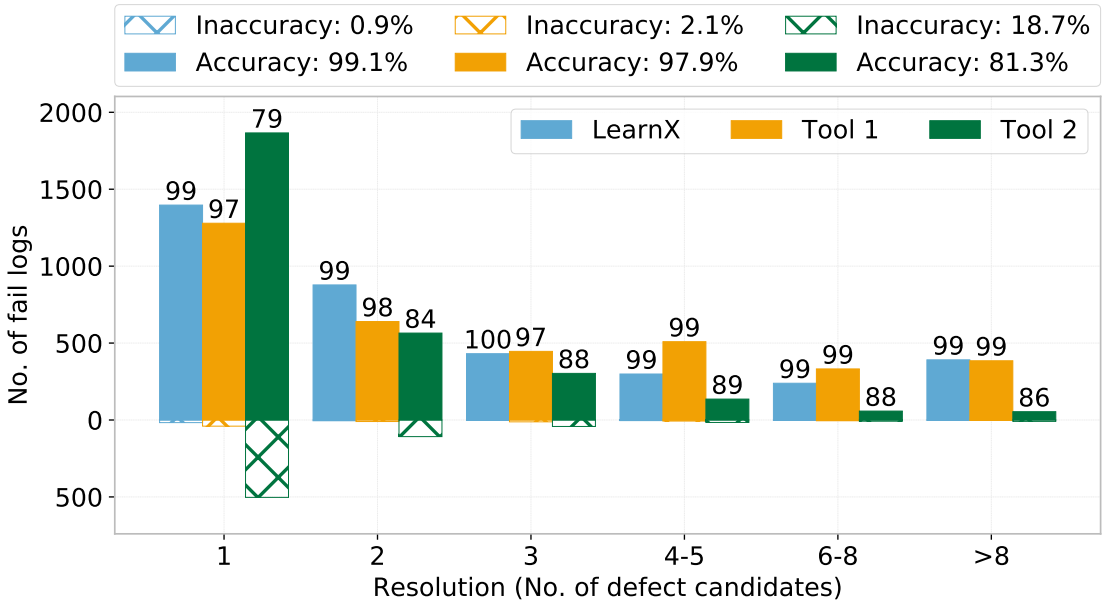


Figure 2.7: Resolution distribution attained by LearnX (Phase 1) for the design "DES".

logs. Phase 1 attains an ideal resolution for 38.5% of fail logs, which is 6.5% better than Tool 1. Additionally, Phase 1 hits a home run for 38.1% of fail logs, which is 9.2% higher than Tool 1. Although Phase 1 returns a single candidate for 40.4% fewer

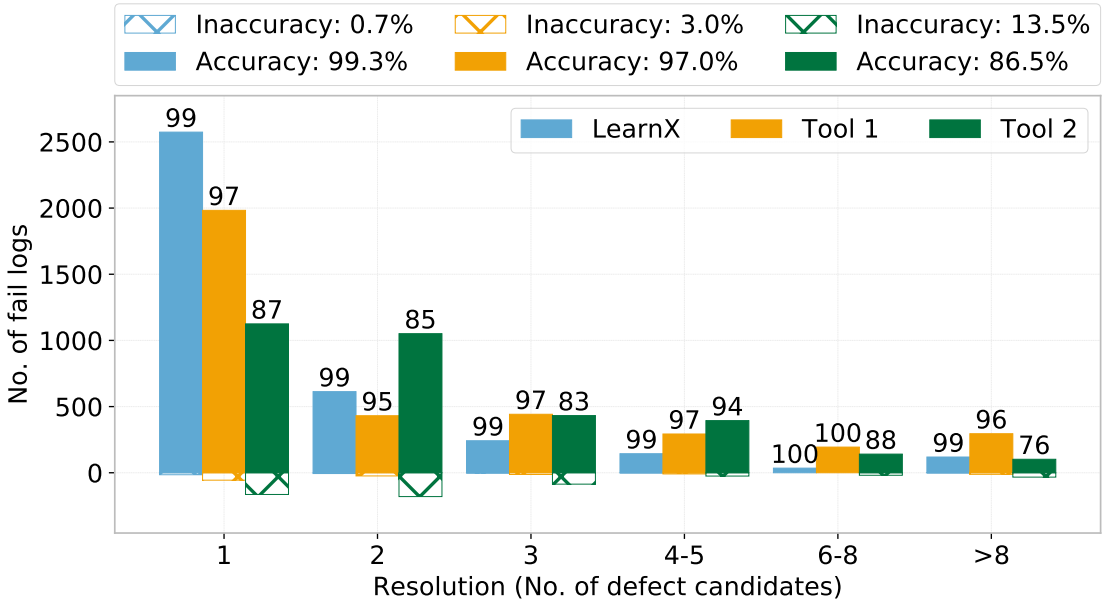


Figure 2.8: Resolution distribution attained by LearnX (Phase 1) for the design "L2B".

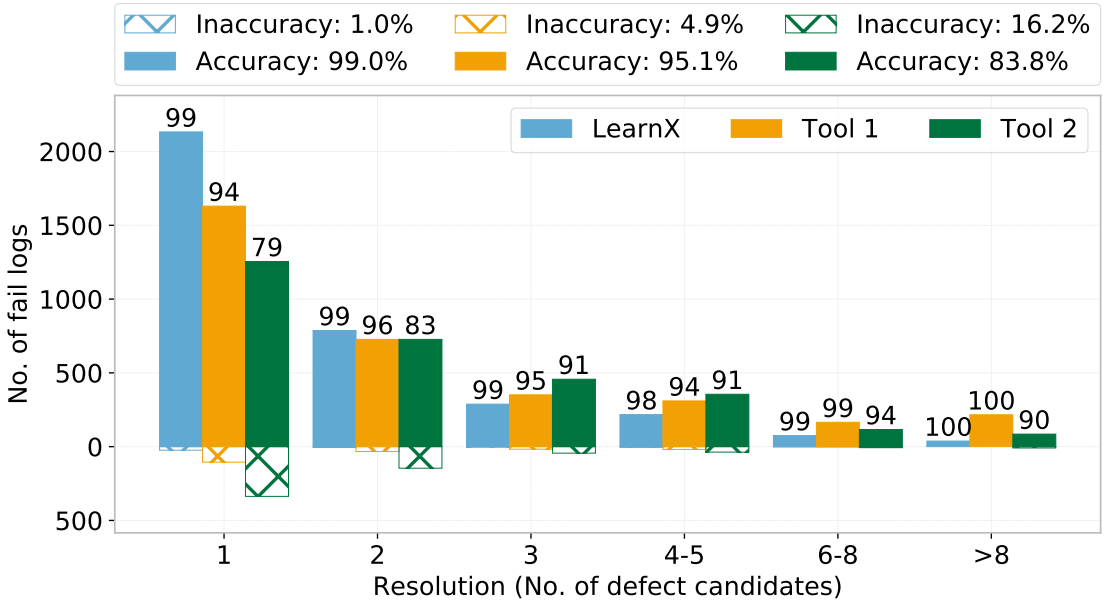


Figure 2.9: Resolution distribution attained by LearnX (Phase 1) for the design "B18".

fail logs when compared to Tool 2, the likelihood of a single candidate being correct is 99.0% for Phase 1 and 78.8% for Tool 2.

3. **L2B**: Figure 2.8 shows that the accuracy achieved by Phase 1 is 99.3% and correctly

diagnoses 2.4% (14.8%) more fail logs than Tool 1 (Tool 2). Phase 1 attains a resolution of one for 69.0% of fail logs, an improvement of 27.1% over Tool 1 and 2X more often than Tool 2. More importantly, Phase 1 delivers a home run for 68.6% of fail logs, while Tool 1 (Tool 2) delivers for 52.9% (30.0%). Moreover, Phase 1 returns at most three candidates for 92.0% of fail logs, which is 17.0% (13.6%) higher than Tool 1 (Tool 2).

4. **B18**: Figure 2.9 shows that the average accuracy for Phase 1 is 99.0%, which is an improvement of 4.1% (18.1%) over Tool 1 (Tool 2). Phase 1 achieves perfect resolution for 60.3% of fail logs with an accuracy of 99.0%. On the other hand, Tool 1 (Tool 2) attains perfect resolution for 48.5% (44.5%) of fail logs, out of which 94.0% (78.9%) are correct.

Table 2.6 reiterates the superior performance of LearnX (Phase 1). Table 2.6 reports the accuracy and *home run* achieved by Phase 1 and commercial diagnosis, per defect type, for each design. For each design, the improvement attained by Phase 1 over Tool 1 and Tool 2 is also shown in the table.

Phase 1 is observed to be especially effective in localizing a bridge defect, in comparison to commercial diagnosis. The minimum accuracy improvement over Tool 1 and Tool 2 is 4.1% and 59.8%, respectively, for AES; the maximum improvement is 20.8% over Tool 1 and 2.4X over Tool 2 for B18. Additionally, the minimum enhancement in *home run* is 23.9% over Tool 1 and 2X over Tool 2 for AES; the maximum enhancement is 28.9% and 11X over Tool 1 and Tool 2, respectively, for L2B.

For open defects, the performance of Phase 1 seems to be similar to (or better than) Tool 1 in terms of accuracy; while the accuracy for L2B is on par, Phase 1 is observed to be the most effective for B18. More importantly, Phase 1 returns an ideal diagnosis more often than Tool 1, where the improvement varies from 9.7% to 44.4%.

Design	Diagnosis method	Accuracy (%)			*Home run* (%)		
		Bridge	Open	Cell	Bridge	Open	Cell
AES	LearnX (Phase 1)	91.9	99.6	100.0	82.9	38.4	39.6
	Tool 1	88.3	99.5	100.0	66.9	35.0	34.5
	Tool 2	57.5	77.6	97.6	41.4	32.5	30.7
	Improvement over Tool 1	4.1	0.1	0.0	23.9	9.7	29.0
	Improvement over Tool 2	59.8	28.4	2.5	101.7	-38.6*	14.8
DES	LearnX (Phase 1)	96.6	99.3	100.0	83.8	22.8	29.5
	Tool 1	91.5	98.9	100.0	67.3	17.8	26.6
	Tool 2	43.4	84.6	94.9	35.7	73.6	26.9
	Improvement over Tool 1	5.6	0.4	0.0	24.5	28.1	10.9
	Improvement over Tool 2	122.6	17.4	5.4	134.7	-69.0†	9.7
L2B	LearnX (Phase 1)	97.8	99.3	100.0	93.7	61.8	60.6
	Tool 1	87.4	99.3	100.0	72.7	51.9	44.6
	Tool 2	49.6	96.8	98.2	8.5	36.0	36.8
	Improvement over Tool 1	11.8	0.0	0.0	28.9	19.1	35.9
	Improvement over Tool 2	97.2	2.6	1.8	1002.4	71.7	64.7
B18	LearnX (Phase 1)	95.8	99.7	100.0	89.9	48.9	53
	Tool 1	79.3	98.2	100.0	70.2	32.2	41.3
	Tool 2	39.8	94.3	96.8	27.6	46.5	32.6
	Improvement over Tool 1	20.8	1.5	0.0	28.1	44.4	28.3
	Improvement over Tool 2	140.7	5.7	3.2	225.7	5.2	62.6

*Although Phase 1 seemingly returns a single correct candidate for 38.6% fewer fail logs than Tool 2, the likelihood of a candidate being correct when a single candidate is reported is 99.0% for Phase 1, and 75.1% for Tool 2. Thus, Phase 1 returns a correct single candidate much more often.

†Although it appears that Phase 1 returns a single correct candidate for 69.0% fewer fail logs than Tool 2, the possibility of a candidate being correct when a single candidate is reported is 97.0% for Phase 1, and 85.1% for Tool 2. Thus, Phase 1 returns a correct single candidate more often.

Table 2.6: Accuracy and *home run* for each defect type achieved by LearnX (Phase 1) and two commercial tools for each of the four designs analyzed.

One point of note is the amount of variation seen in the number of ideal diagnoses returned by Phase 1 and Tool 2 for open defects. While Tool 2 appears to be better for AES and DES, Phase 1 is more effective for L2B and B18. However, the accuracy for Tool 2 is much lower. Further investigation reveals that a significant number of diagnoses are incorrect when a single candidate is returned by Tool 2. In other words, the probability of a single candidate being correct is lower for Tool 2. Specifically, for AES, Tool 2 diagnoses 75.1% of fail logs correctly with a resolution of one, while Phase 1 diagnoses 99.0%. Similarly, for DES, among the fail logs diagnosed with perfect resolution, 85.1% of fail logs are accurate when Tool 2 is used while 97.0% of fail logs are accurate when Phase 1 is used. A subsequent failure analysis method can be misguided if a supposedly ideal diagnosis is incorrect. As a result, significant PFA resources can be exhausted/misused, likely hindering yield learning.

For cell defects, Phase 1 is more productive than Tool 1 and Tool 2 as well. In addition to correctly locating a defect for each fail log, Phase 1 hits a home run more often than Tool 1 and Tool 2 for each design. The range of the improvement in *home run* obtained by Phase 1 over Tool 1 is 10.0% to 35.9%, and over Tool 2 is 9.7% to 64.7%.

2.3.2.2 Phase 2

Figures 2.10 through 2.13 show the histograms of the resolution achieved by Phase 2 for the four designs examined. Observations specific to each design are as follows.

1. **AES**: It can be observed from Figure 2.10 that Phase 2 correctly locates a defect for 97.7% of fail logs, while Tool 1 (Tool 2) correctly locates a defect for 88.3% (62.7%) of fail logs. Phase 2 attains an ideal resolution for 98.4% of fail logs, which is 2.2X times Tool 1 and 40.8% better than Tool 2. Additionally, Phase 2 hits a home run for 96.4% of fail logs, which is 2.4X (2.1X) more often than Tool 1 (Tool 2). Further investigation reveals that Phase 2 returns at most four candidates for all the fail logs.

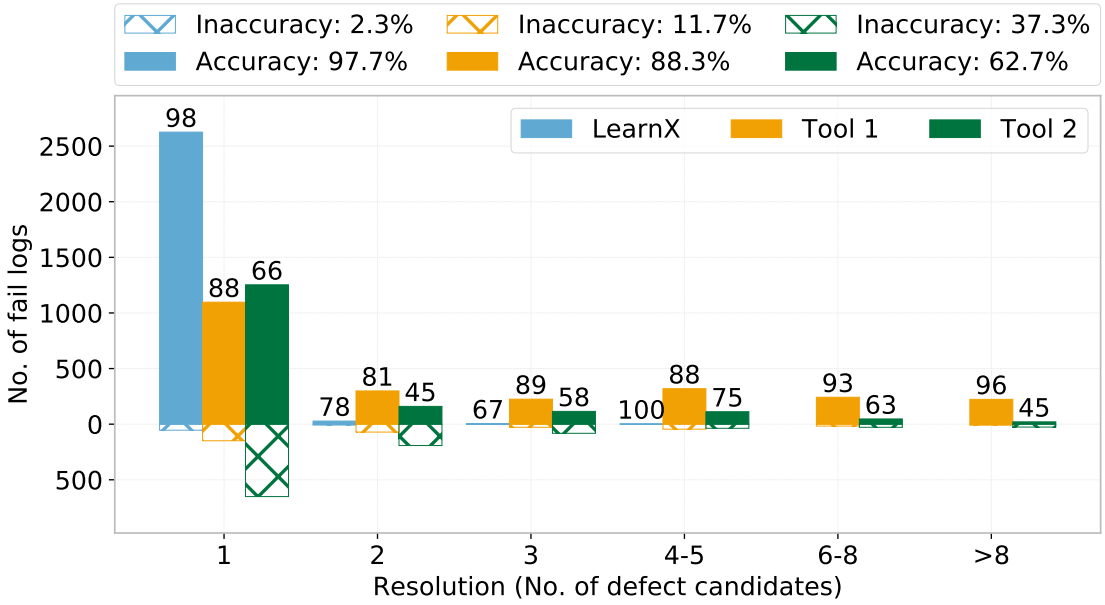


Figure 2.10: Resolution distribution attained by LearnX (Phase 2) for the design "AES".

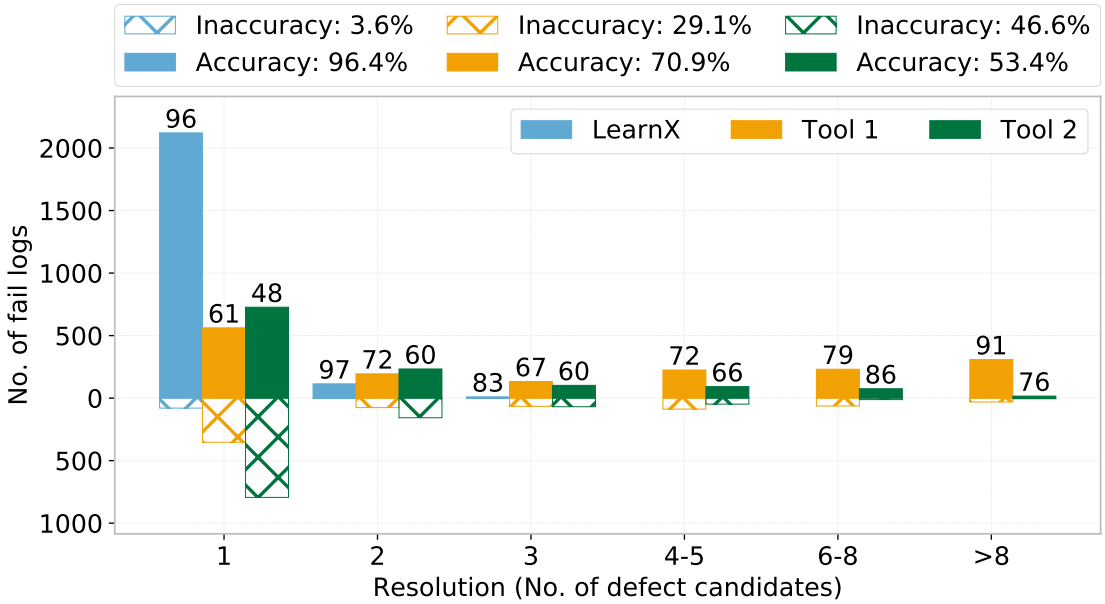


Figure 2.11: Resolution distribution attained by LearnX (Phase 2) for the design "DES".

2. **DES**: Figure 2.11 reveals that the average accuracy for LearnX (Phase 2) is 96.4%, which is 36.0% (80.5%) more than Tool 1 (Tool 2). In addition, Phase 2 attains perfect resolution for 94.5% of fail logs, of which 96.3% identify the correct candidate. On the

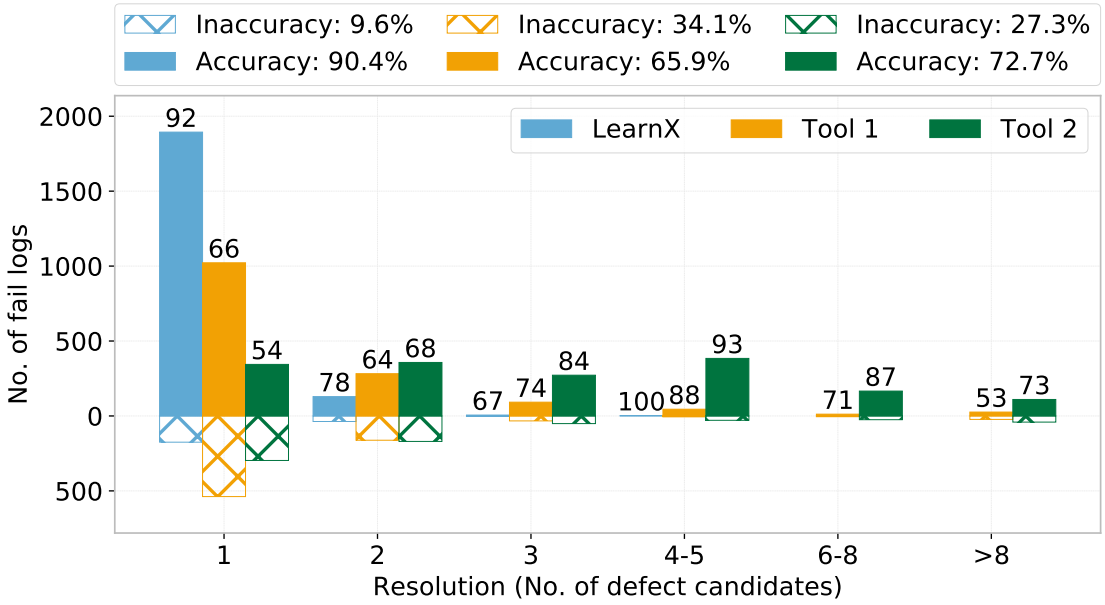


Figure 2.12: Resolution distribution attained by LearnX (Phase 2) for the design "L2B".

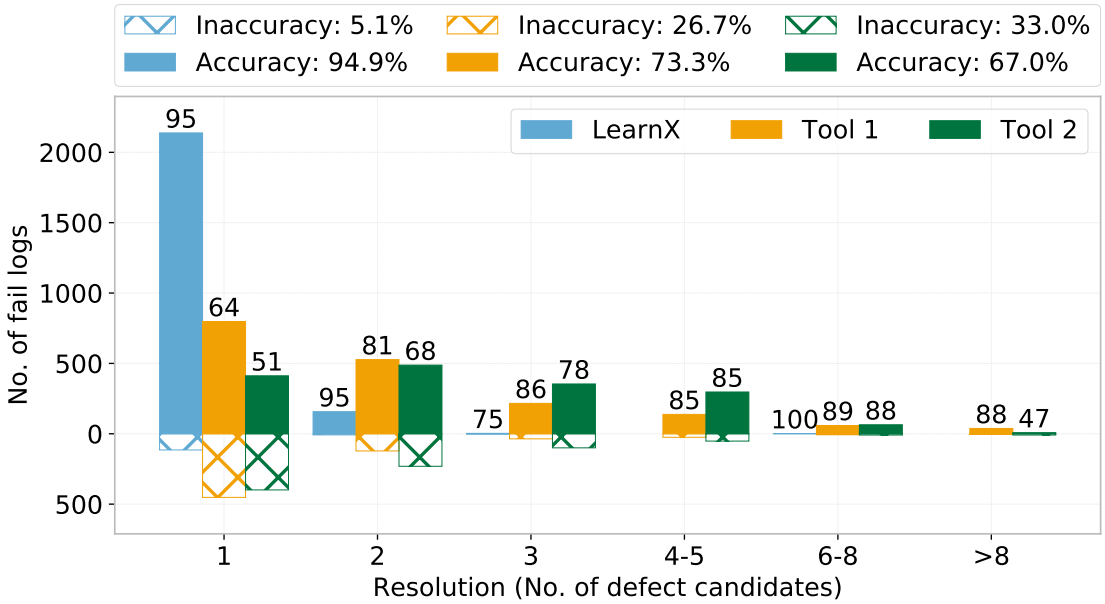


Figure 2.13: Resolution distribution attained by LearnX (Phase 2) for the design "B18".

other hand, Tool 1 (Tool 2) is 61.3% (47.8%) correct among the defects diagnosed with perfect resolution. Moreover, Phase 2 delivers a home run (i.e., returns an ideal diagnosis) for 91.0% of fail logs, which is 3.7X (2.9X) times Tool 1 (Tool 2). Further

analysis shows that Phase 2 returns at most three candidates for all the fail logs.

3. **L2B**: Figure 2.12 shows that the average accuracy for Phase 2 is 90.4%, which is an improvement of 37.2% (24.3%) over Tool 1 (Tool 2). LearnX achieves an ideal resolution for 92.1% of fail logs with an accuracy of 91.5%. On the other hand, Tool 1 (Tool 2) attains perfect resolution for 69.5% (28.5%) of fail logs, out of which 65.5% (53.7%) identify the correct candidate. Additionally, Phase 2 reports at most five candidates for all the fail logs.

4. **B18**: Figure 2.13 shows that the accuracy achieved by Phase 2 is 94.9% and correctly diagnoses 29.5% (41.6%) more fail logs than Tool 1 (Tool 2). Phase 2 attains a resolution of one for 93.0% of fail logs, which is an improvement of 80.2% over Tool 1 and 2.8X more often than Tool 2. More importantly, Phase 2 delivers a *home run* for 88.2% of fail logs, while Tool 1 (Tool 2) delivers for 32.9% (17.0%). Moreover, Phase 2 returns at most three candidates for all the fail logs (except one, where the resolution is six).

Table 2.7 reiterates the effectiveness of the combination of the X-fault model and machine learning to pinpoint the correct candidate in Phase 2. Table 2.7 shows the accuracy and *home run* attained by LearnX (Phase 2) and commercial diagnosis, per defect type, for each design. For each design, the enhancement of LearnX (Phase 2) over Tool 1 and Tool 2 is also reported in the table.

It is evident from Table 2.7 that the improvement in accuracy and *home run* over commercial diagnosis is significant for both defect types[9].

For bridge defects, the accuracy improvement obtained by Phase 2 over Tool 1 varies from 23.2% to 91.1%. The minimum improvement in accuracy over Tool 2 is 62.7% (for L2B); in addition, Phase 2 correctly diagnoses a bridge defect at most 2.6X times Tool 2

[9]Each cell defect is diagnosed successfully in Phase 1 and hence, none is analyzed in Phase 2.

		Accuracy (%)		*Home run* (%)	
Design	Diagnosis method	Bridge	Open	Bridge	Open
	LearnX (Phase 2)	96.6	98.5	95.5	97.2
	Tool 1	78.4	97.0	14.1	63.2
	Tool 2	37.4	85.1	21.1	68.0
	Improvement over Tool 1	23.2	1.5	577.3	53.3
AES	Improvement over Tool 2	158.3	15.7	352.6	42.9
	LearnX (Phase 2)	94.3	98.6	91.7	90.3
	Tool 1	53.0	90.2	10.5	38.7
	Tool 2	38.4	69.6	25.0	37.8
	Improvement over Tool 1	77.9	9.3	773.3	133.3
DES	Improvement over Tool 2	145.6	41.7	266.8	138.9
	LearnX (Phase 2)	90.6	90.3	89.7	79.0
	Tool 1	47.4	84.3	21.7	69.3
	Tool 2	55.7	89.7	5.7	24.9
	Improvement over Tool 1	91.1	7.1	313.4	14.0
L2B	Improvement over Tool 2	62.7	0.7	1473.7	217.3
	LearnX (Phase 2)	94.7	95.1	93.1	83.4
	Tool 1	60.6	86.1	15.0	51.0
	Tool 2	40.2	94.0	7.2	26.9
	Improvement over Tool 1	56.3	10.5	520.7	63.5
B18	Improvement over Tool 2	135.6	1.2	1193.1	210.0

Table 2.7: Accuracy and *home run* for each defect type achieved by LearnX (Phase 2) and two commercial tools for each of the four designs analyzed.

(for AES). The number of ideal diagnoses reported by Phase 2, in comparison to commercial diagnosis, is more impressive. Phase 2 hits a home run at least 4.1X (3.7X) times Tool 1 (Tool 2), and at most 8.7X (15.7X) times Tool 1 (Tool 2).

For open defects, the range of the enhancement in accuracy over Tool 1 (Tool 2) is 1.5% to 10.5% (0.7% to 41.7%). Phase 2 delivers at least 14.0% (42.9%) more home runs than Tool 1 (Tool 2), and up to 2.3X (3.2X) more often than Tool 1 (Tool 2).

Furthermore, Table 2.7 reveals that Phase 2 is relatively more effective in identifying a bridge defect than an open defect. Specifically, Phase 2 correctly diagnoses 57.1% (1.2X) more fail logs associated with a bridge defect than Tool 1 (Tool 2). On the other hand, the accuracy (that is averaged over the four designs analyzed) for Phase 2 for open defects is 95.6%, an improvement of 7.0% (13.0%) over Tool 1 (Tool 2). In addition, Phase 2 delivers a home run 6X more often than Tool 1 and Tool 2 for bridge defects, and 57.1% (2.1X) more than Tool 1 (Tool 2).

2.3.2.3 Results Summary

Experiment results presented in Sections 2.3.2.1 and 2.3.2.2 demonstrate the superior performance of Phase 1 and Phase 2, respectively. This section summarizes the overall effectiveness of LearnX in comparison to commercial diagnosis, irrespective of its phases.

Figure 2.14 illustrates the accuracy achieved by LearnX for the four designs examined and compares it with commercial diagnosis. The horizontal axis shows the accuracy for each design and for each diagnosis approach. The vertical axis represents each design. Four sets of values/percentages, one for each design, show the improvement in accuracy attained by LearnX over Tool 1 and Tool 2.

It can be seen from Figure 2.14 that the maximum improvement over Tool 1 is observed for B18 (specifically, 12.7%), and over Tool 2 for DES (specifically, 39.0%). The accuracy for LearnX is at least 96.0% (for L2B) and at most 98.0% (for DES). On the other hand, the accuracy for Tool 1 ranges from 85.3% to 93.2%, and Tool 2 ranges from 70.5% to 81.3%.

Figure 2.15 highlights the number of ideal diagnoses reported by LearnX and commercial diagnosis for the four designs examined. The *home run* attained by LearnX varies from 58.7% to 74.5%. The improvement in the number of ideal diagnoses over Tool 1 (Tool 2) is maximum for DES (L2B), where LearnX returns an ideal diagnosis for 91.2% more fail logs when compared to Tool 1, and up to 3X times Tool 2.

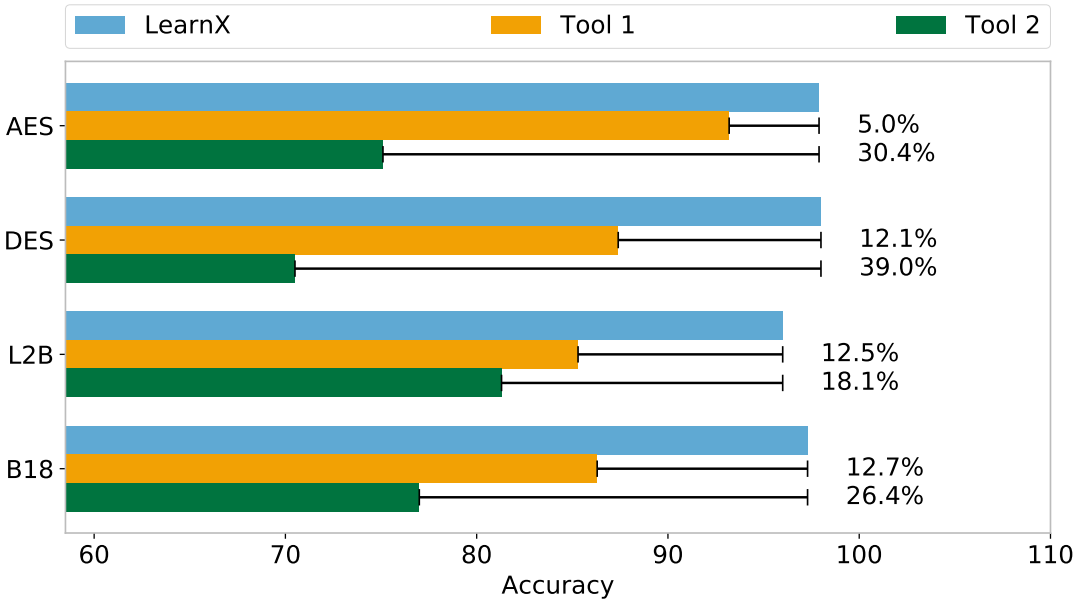


Figure 2.14: Accuracy achieved by LearnX and commercial diagnosis.

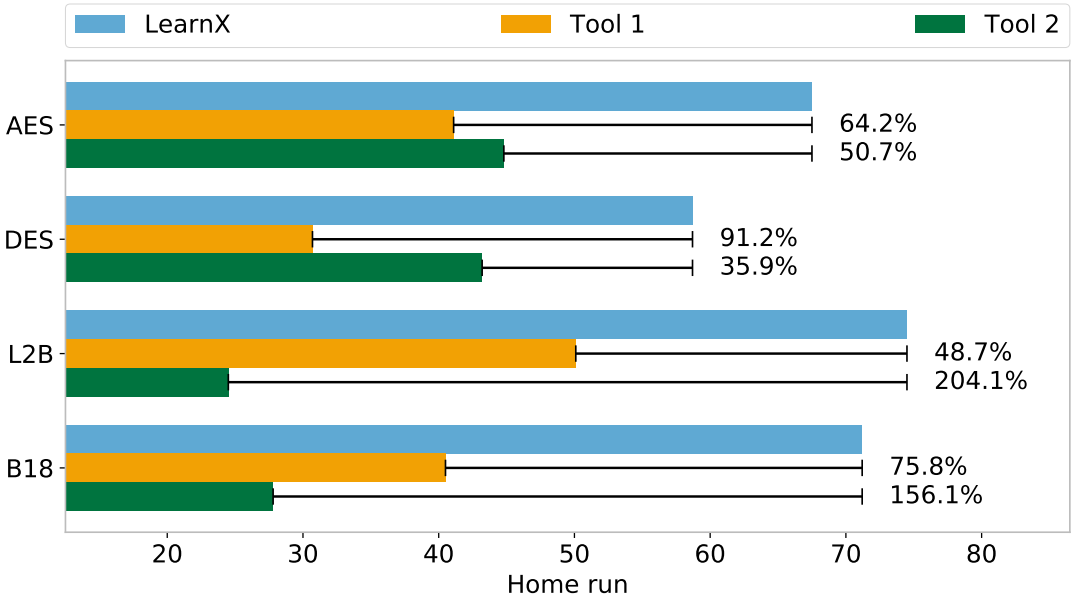


Figure 2.15: *Home run* achieved by LearnX and commercial diagnosis.

High accuracy and *home run* imply that there is less ambiguity introduced in the diagnosis results. In other words, there is less uncertainty of whether a candidate represents a defect or

not. Consequently, volume diagnosis can estimate a failure root cause with more assurance, which could lead to higher PFA success rate, and possibly, accelerate yield learning.

When the runtime (wall clock time or elapsed real time, to be precise) of LearnX is compared with commercial diagnosis, it is observed that, on average, LearnX is 27.4% slower than Tool 1 and 44.8% faster than Tool 2. This comparison, however, does not include the one-time cost (per design) of training the machine learning models. For instance, for the largest design, AES, it takes 2.2 hours to create all the four machine learning models, out of which, a majority of time (89.4%) is consumed for generating the training datasets. That being said, it should be noted that LearnX is implemented here to demonstrate its effectiveness (in terms of the diagnostic metrics discussed in Section 2.3.1) instead of its runtime efficiency. One of the future goals is to optimize the runtime of LearnX.

2.3.3 Silicon Experiment

The significance of LearnX is further demonstrated using an industrial design fabricated in an advanced process node. The setup for the conducted experiment is described in Section 2.3.3.1. It conveys basic information about the design and the silicon failure data that is available to study the effectiveness of LearnX. The results of the experiment are discussed in Section 2.3.3.2 to explore how LearnX fares in the real world.

2.3.3.1 Setup

LearnX is evaluated using an industrial chip that is manufactured in a 14nm technology. The chip measures 12 mm^2 in size and is divided into 12 partitions. Each partition consists of approximately 3.5 million gates. Each partition is tested using approximately 825 test patterns[10] that have been generated using a commercial ATPG software. Fail logs corresponding to 2,400 failing chips are analyzed in this experiment. Thirty-six failing chips have

[10]The size of the test set obtained for each partition ranges from 806 to 857.

been PFA'ed.

The LearnX flow illustrated in Figure 2.2 is applied to each fail log. Each fail log is also analyzed by commercial diagnosis[11], where only the top-scoring candidates are considered as its output. Each diagnosis technique is gauged using the diagnosis metrics defined in Section 2.3.1. The resolution for each diagnosis method is calculated for each failing chip, while the accuracy and *home run* are also compared for the 36 PFA'ed chips.

2.3.3.2 Results

Figure 2.16 reports the distribution of resolution produced by LearnX and commercial diagnosis for 2,400 fail logs. The horizontal axis shows the resolution and the vertical axis shows the number of fail logs. It is observed from Figure 2.16 that among 2,400 fail logs, LearnX returns an ideal resolution for 1,127 (46.9%) fail logs. On the other hand, commercial diagnosis returns a resolution of one for 576 (24.0%) fail logs, which is almost half of LearnX. In addition, LearnX reports a resolution of at most three for 1,805 (75.1%) fail logs, which is 59.0% more than commercial diagnosis.

Figure 2.17 reports the resolution for each fail log obtained by each diagnosis method. The figure is divided into two plots to better visualize the variation in the resolution. The x-axis represents the fail logs and the y-axis shows the attained resolution in each plot. Figure 2.17(a) sorts the fail logs by commercial diagnosis resolution, and includes the fail logs whose commercial diagnosis resolution is less than 15. Figure 2.17(b) sorts the fail logs for which commercial diagnosis returns at least 15 candidates. Note that the scale of both the axes in both the plots is different. The x-axis varies from 0 to 2086 in Figure 2.17(a), and from 2087 to 2399 in Figure 2.17(b). The y-axis ranges from 1 to 14 in Figure 2.17(a), and from 15 to 319 in Figure 2.17(b).

It is clearly seen from Figure 2.17 that the resolution for LearnX is better than or identical

[11]The available data conforms to Tool 1 and hence, the diagnosis quality attained by LearnX is compared with Tool 1 only.

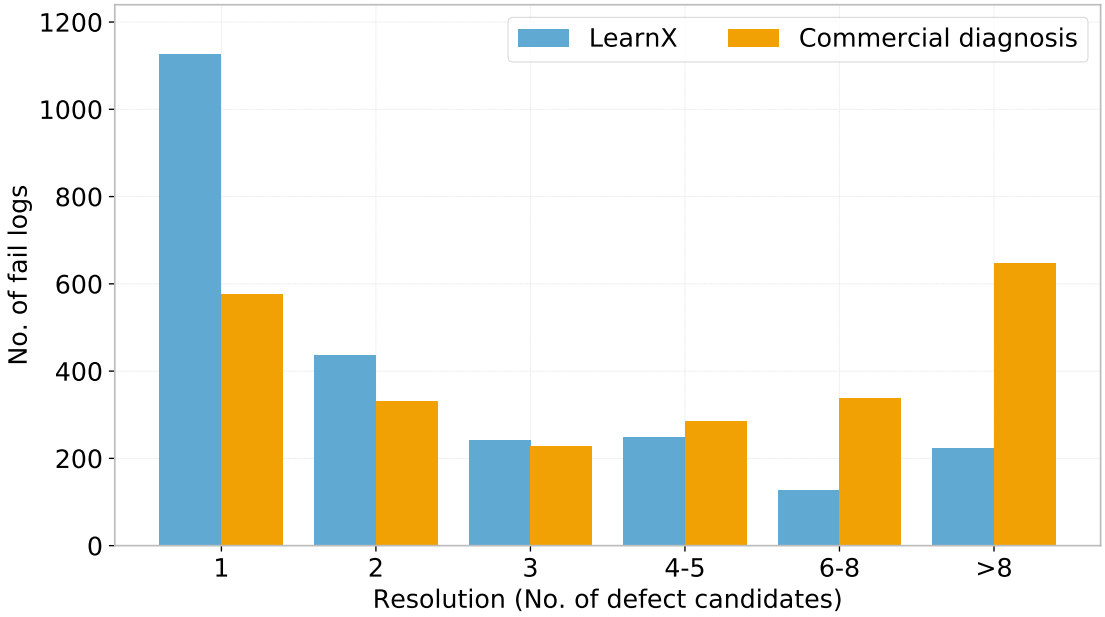


Figure 2.16: Resolution distribution attained by LearnX for 2,400 silicon fail logs.

to commercial diagnosis. Specifically, LearnX returns 4.3 candidates per fail log, on average, while commercial diagnosis returns 11.4 candidates. This means that LearnX reports 7.1 or 62.3% fewer candidates per fail log. The maximum resolution obtained for commercial diagnosis is 319, which is three times the maximum resolution for LearnX. Figure 2.17 further shows that LearnX enhances the resolution for 1,716 (71.5%) fail logs. In addition, LearnX improves the resolution by 40.8%, if averaged over all the fail logs; the average improvement goes up to 57.1%, if the resolution improvement is averaged over the fail logs where resolution is indeed enhanced.

Among 2,400 failing chips, 36 failing chips are PFA'ed. Out of these 36 failing chips, 19 are suspected to have a single defect. The rest of the 17 chips are analyzed by MD-LearnX in Section 3.3.3.

PFA confirms that LearnX (and commercial diagnosis) correctly pinpoints the correct candidate for each of the 19 failing chips. More importantly, LearnX achieves superior resolution and *home run* without sacrificing accuracy. Figure 2.18 reports the resolution

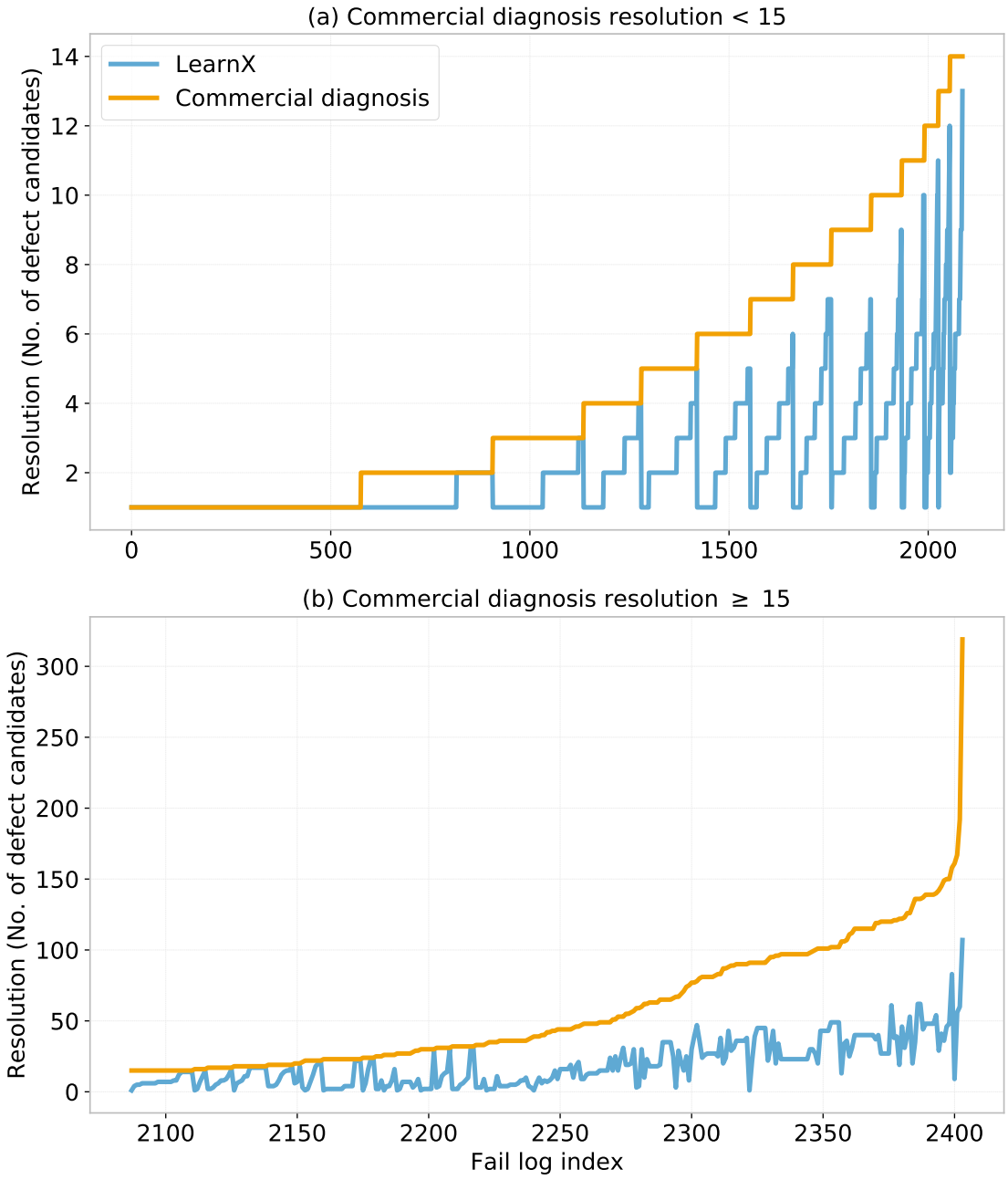


Figure 2.17: Resolution comparison for LearnX and commercial diagnosis for each silicon fail log. The fail logs are divided in two groups based on commercial diagnosis resolution for an aesthetic reason: (a) fail logs with commercial diagnosis resolution of less than 15, and (b) fail logs with commercial diagnosis resolution of at least 15.

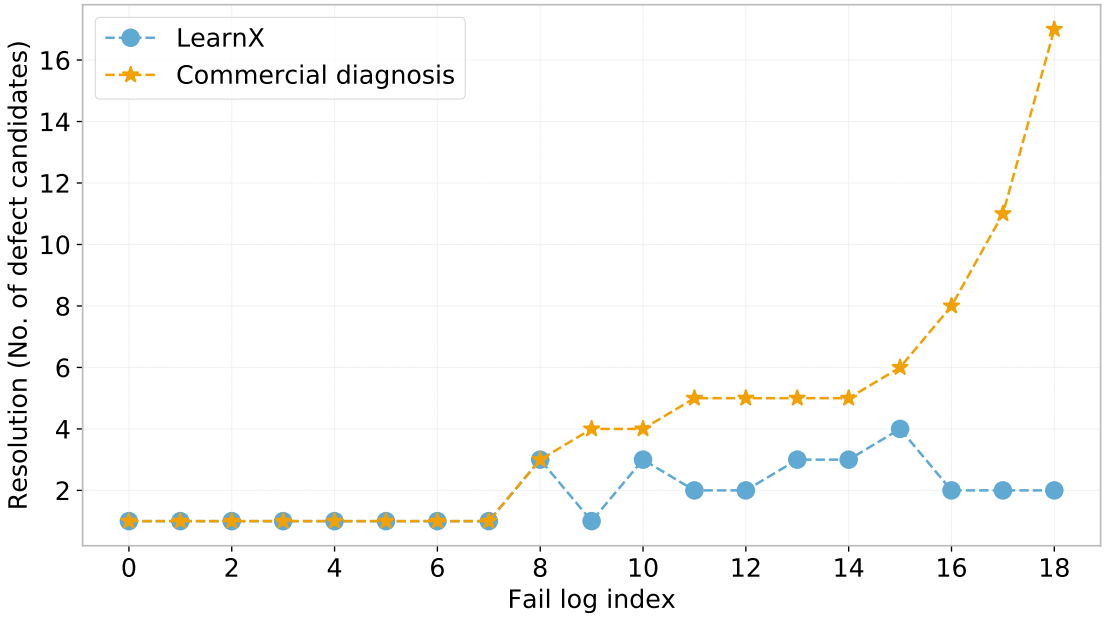


Figure 2.18: Resolution comparison for LearnX and commercial diagnosis for each failing chip that is PFA'ed and suspected to have a single defect.

obtained by each diagnosis method for each of the 19 failing chips.

Figure 2.18 reveals that LearnX returns fewer defect candidates than commercial diagnosis for 10 (52.6%) fail logs, while returning the same candidates for other fail logs, without losing accuracy. On average, LearnX identifies 1.8 candidates per fail log, which is 2.4 fewer candidates than commercial diagnosis. The maximum resolution attained by commercial diagnosis is 17, which is more than four times the maximum resolution for LearnX. In addition, LearnX improves the resolution by 30.4%, if averaged over all the fail logs, and 57.8% if averaged over the ones where resolution is enhanced. Furthermore, LearnX hits a home run for 52.9% of fail logs, i.e., 12.6% more fail logs, than commercial diagnosis.

2.4 Conclusion

A single-chip diagnosis methodology called LearnX is described that characterizes a defect with respect to its physical location and behavior. LearnX is a two-phase methodology. In the first phase, defects that mimic traditional fault models are identified using a set of rules derived from test data. In the second phase, a machine learning classifier is created that differentiates correct and incorrect candidates by learning from the candidate features extracted from the test data. The features are derived by comparing the tester response with the fault simulation response of a candidate. In contrast to a traditional diagnosis approach, the X-fault model, in addition to the stuck-at fault model, is employed to capture a comprehensive depiction of the impact of a defect on the circuit.

Several experiments are conducted to evaluate the performance of LearnX. Simulation-based experiments are carried out for four different designs with 6,000 faulty circuits each (that are created using a variety of realistic defect behaviors including byzantine bridges and opens). The experiments demonstrate the potential of LearnX. LearnX achieves an average accuracy of 97.3%, while two state-of-the-art commercial diagnosis tools achieve 88.1% and 76.0%. Thus, LearnX identifies the correct candidate for at least 9.2% more fail logs than commercial diagnosis.

In addition, LearnX returns a single candidate for 70.1% fail logs, an improvement of 46.7% and 41.5% over commercial tools. When the number of diagnoses with high resolution (i.e., when at most three candidates are reported) are analyzed, it is observed that LearnX returns a resolution of three or less for a majority of fail logs (specifically, 89.3%). More importantly, LearnX hits a home run (i.e., when a single candidate is identified correctly) significantly more often than commercial diagnosis. LearnX delivers a home run for 67.9% of fail logs, which is 67.5% better than one commercial tool and almost double the second commercial tool.

Moreover, significance of LearnX is substantiated by diagnosing 2,400 silicon fail logs from a design fabricated in an advanced process technology. It is revealed that LearnX returns an ideal resolution for 46.9% fail logs, which is almost twice as often as commercial diagnosis. Additionally, LearnX returns 7.1 fewer candidates per fail log, on average. LearnX enhances the resolution for 71.5% of fail logs, and achieves an average resolution improvement of 40.8%.

The effectiveness of LearnX is further corroborated by inspecting 19 failing chips that are PFA'ed. **LearnX is able to correctly locate a defect in each failing chip, while reporting fewer candidates than state-of-the-art commercial diagnosis.** LearnX returns fewer defect candidates than commercial diagnosis for 52.6% fail logs. On average, LearnX identifies 2.4 fewer candidates per fail log. Furthermore, LearnX delivers a home run for 52.9% of fail logs, which is 12.6% better than commercial diagnosis.

High diagnosis resolution and accuracy mean that a subsequently applied volume diagnosis approach will generate a more precise pareto of probable yield loss failure mechanisms, thus likely enabling rapid yield learning.

Chapter 3

MD-LearnX: A Deterministic-Statistical Multiple Defect Diagnosis Methodology

Software diagnosis is a process of locating and characterizing a defect in a failing chip. It is the cornerstone of failure analysis that consequently enables yield learning and monitoring. Chapter 2 elucidates LearnX, a diagnosis methodology that focuses on pinpointing the location of a defect (and its physical fault type) in a failing chip. LearnX is a two-phase physically-aware approach. In the first phase, a defect that resembles a traditional fault model is identified through a set of strict rules. In the second phase, supervised machine learning is used to determine the correct candidate.

Results presented in Section 2.3 demonstrate the superior performance of LearnX for single-defect diagnosis when compared with leading-edge commercial diagnosis. However, with decreasing feature sizes, and increasing interconnect density and manufacturing complexity, more chips are failing due to multiple defects, particularly when systematic defects (that arise from unforeseeable process-design interactions) are the dominant yield limiters

Chip type	Process node (nm)	Failing chips diagnosed with multiple defects (%)	No. of failing chips
High-volume chip	55	11.7	1,201
Test chip	14	15.6	1,375
Test chip	28	29.8	1,952
Test chip	14	32.2	11,727
High-volume chip	130	32.6	328
High-volume chip	90	35.5	9,536
Test chip	110	37.7	5,416
High-volume chip [136, 137]	130	41.0	453
High-volume chip	–	49.8	353
Test chip	28	53.9	167
[318]	55	60.0	209

Table 3.1: Summary of silicon test data from chips manufactured across various process nodes and organizations highlighting the percentage of failing chips affected by multiple defects.

(either in the early stages of yield learning or due to yield excursion).

To evidence the importance of multiple-defect diagnosis, silicon test data from tens of thousands of failing chips manufactured by various organizations in process nodes ranging from 130nm to 14nm is collected and analyzed. The results are summarized in Table 3.1. The first column ("Chip type") specifies whether the silicon data is obtained for a test chip or high-volume manufacturing chip. The second column shows the process node in which a chip is fabricated. Each fail log that is obtained is examined using commercial diagnosis. The third column shows the percentage of failing chips that are suspected to be affected with more than one defect. The fourth column ("No. of failing chips") specifies the size of the dataset analyzed. The table is sorted by the third column in ascending order of the percentage of failing chips diagnosed with more than one defect.

Table 3.1 reveals that the frequency of a failing chip diagnosed with multiple defects is significant. Specifically, 33.1% of failing chips exhibit behavior consistent with multiple

defects, on average. Moreover, for the dataset corresponding to the last row, diagnosis suggests that there is a strong indication of multiple defects for 60.0% of the failing chips. Increasing number of chips with multiple defects thus warrants the need of a diagnosis technique that is capable to adequately identify multiple defects.

To that end, a single-chip diagnosis methodology that we term MD-LearnX is developed and described in this chapter. It builds on LearnX (hence the name "MD-LearnX", where "MD" indicates multiple defects) to effectively tackle the task of locating more than one defect, and in turn, aid in accelerating the design and process development.

MD-LearnX is a three phase, physically-aware diagnosis approach. The first phase identifies a defect that mimics well-known fault models. The second and the third phases utilize the X-fault model and machine learning to identify correct candidates. While the second phase focuses on identifying multiple non-interacting defects, the third phase is proficient in finding multiple interacting defects.

LearnX/MD-LearnX serves as the foundation for the subsequent diagnosis methods, called PADLOC and NOIDA, developed in this dissertation. The goal of PADLOC and NOIDA is to physically localize, and logically characterize a back-end and a front-end defect, respectively. The performance of LearnX/MD-LearnX, thus, has a cascade effect on the effectiveness of PADLOC and NOIDA – inadequate defect localization by LearnX/MD-LearnX causes incompetent defect characterization by PADLOC and NOIDA. Thus, the quality of defect localization achieved by LearnX/MD-LearnX is critical to the success of our overall diagnosis methodology. Results reported in Section 2.3 show that LearnX outperforms state-of-the-art commercial diagnosis for single defects. MD-LearnX aims to replicate the same for multiple defects.

The rest of the chapter is organized as follows. Section 3.1 reviews prior work related to multiple-defect diagnosis and motivates MD-LearnX. Section 3.2 provides a detailed overview of the three phases of MD-LearnX to locate multiple defects in a failing chip. The efficacy of

LearnX is demonstrated via several experiments that are described in Section 3.3. Finally, Section 3.4 summarizes the overall contributions of this work.

3.1 Prior Work

Several methods have been suggested over the years to examine multiple defects. Compared to the diagnosis of a single defect, characterizing each defect in a chip affected by multiple defects is challenging, primarily due to two reasons. First, erroneous values propagating from more than one defect location can interact with each other, resulting in either error masking (where one error blocks the propagation of another) or error unmasking (where one error assists the propagation of another). Figure 3.1 illustrates error masking and unmasking. In Figure 3.1(a), the error on input B of a 2-input AND cell blocks the propagation of error manifested at input A. In Figure 3.1(b), the error on input B assists the propagation of error manifested at input A.

The second challenge in multiple-defect diagnosis is that the solution search space is exponential in design size and defect multiplicity (unknown beforehand), which makes finding an optimum solution extremely difficult.

Multiple-defect diagnosis approaches in the literature can be divided into different categories. The first category of methods (e.g., [115, 131–144, 319]) perform per-test diagnosis.

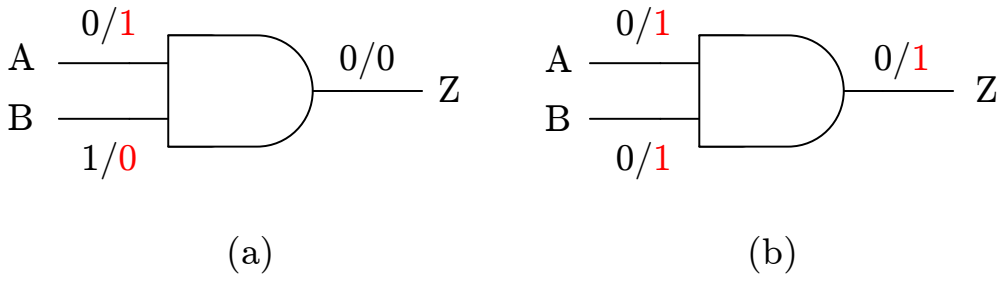


Figure 3.1: Example to illustrate (a) error masking and (b) unmasking.

Such methods have been reviewed in Section 2.1. Essentially, those methods rely on identifying a defect via failing patterns that can be explained by a single stuck-at fault. Such failing patterns are called SLAT (Single-Location-At-a-Time) [136, 137] patterns, or type-1 patterns [320]. In those methods, an initial set of candidates that explain each type-1 pattern is first derived. Then, a final set of candidates that collectively explain all the type-1 patterns is identified using a set-cover and/or an ad-hoc candidate-scoring technique. The drawback of those methods is that they are successful only when multiple defects affect a failing chip such that only one of the defect locations is sensitized for a failing pattern. However, such patterns can be limited and/or find an incorrect location due to a considerable number of interactions among errors manifesting at multiple defect locations.

In addition to considering type-1 patterns, the approaches presented in [174, 318, 321–325] also take into account failing patterns where (binary) errors propagating from multiple locations do not interfere with each other. Such failing patterns are referred to as type-2 patterns [320]. Thus, those methods restrict themselves in identifying multiple, non-interacting defects. However, as defect multiplicity increases, the likelihood of error masking or unmasking increases. As a consequence, the number of type-1 and type-2 patterns may decrease and may not be sufficient for an effective diagnosis.

The approach presented in [174,324,325] goes a step further. That approach begins with deriving a set of candidates (stuck-at faults, in particular) that collectively explains each type-1 and type-2 pattern. Candidate covers of minimum size are then analyzed with type-3 patterns. (Any failing pattern that is neither type-1 nor type-2 is designated as a type-3 pattern.) A scoring procedure is then designed to rank the covers.

The second category of methods follows an incremental approach to diagnose multiple defects [320, 326–329]. In incremental diagnosis, stuck-at faults that explain each type-1 pattern are first identified. Each stuck-at fault in then injected in the circuit, one by one, to produce a modified circuit that corresponds to each identified fault. The procedure to find

type-1 patterns and the corresponding set of faults that explain each type-1 pattern is again applied to each modified circuit. The process is repeated until all the failing patterns have been explained. Various heuristics have been put forward to accelerate the aforementioned approach. For example, each stuck-at fault in an iteration can be ranked based on the number of failing and passing patterns it explains; the highest ranked fault can be then selected to alter the circuit for the next iteration. However, an inaccurate solution may be derived if an incorrect fault is chosen at any iteration. In addition, incremental diagnosis performs numerous multiple fault simulations. As a result, an incremental diagnosis method does not scale well with design size and defect multiplicity.

Contrary to the extensive enumeration and simulation of all possible sets of faults (example methods that implement such an approach include [330–332]), the third category of methods guide their effort in exploring the exponential search space via optimization methods [333–336]. For example, a Particle Swarm Optimization (PSO) algorithm is employed in [333, 334] to locate multiple defects. However, PSO does not guarantee that an optimal solution will be identified. In addition, numerous multiple fault simulations are still needed to search the solution space and hence, such methods are impractical for large industrial designs.

The fourth category of methods avoids explicit fault simulation to identify multiple defects [337–343]. For each failing pattern, each candidate location in a design is scored based on its ability to propagate an error to a design output while considering error masking and unmasking. Each method then, iteratively and greedily, selects the most likely set of defect locations based on a candidate ranking procedure.

For example, the approach presented in [337] tracks the number of failing outputs explained by each fault in the circuit for each failing pattern using a procedure similar to path tracing, while taking error masking and unmasking into account. The fault with the highest score, i.e., the fault that explains the most number of failing outputs across all failing

patterns, is then selected as a candidate. The scores of the remaining faults are updated to find a candidate that can explain the most number of remaining unexplained failing outputs. This process continues until all the outputs have been explained. However, in any iteration, if there are multiple faults that have the same score, they are analyzed separately. As a result, the number of such analyses can increase exponentially in each iteration, rendering that approach nonviable because of large time and memory overhead, and poor diagnostic resolution. Furthermore, the approach presented in [337] does not utilize passing patterns or layout information to improve the quality of diagnosis.

The work of [341–343] avoids explicit simulation of multiple faults as well. It begins with identifying potential defective signals using path tracing [108–110]. Because path tracing is conservative, some candidates derived from path tracing a failing pattern may cause a passing output to fail in spite of the other defective signals in the circuit. Thus, that approach devises a method to eliminate those candidates by utilizing the X-fault model. The work of [344] uses a similar technique but is limited to only stuck-at faults.

The approach of [341–343] then analyzes and compares the fault effects of each candidate defect on the failing circuit. It examines the error propagation paths of each candidate defect, while considering the errors manifesting at other locations. Minimally-sized groups of candidates are then selected to explain the failing patterns, which are subsequently ranked based on their responses to a passing pattern. However, the diagnosis time and the memory usage needed to analyze each possible error propagation path increase rapidly with design size and defect multiplicity.

The works of [345–349] also utilize X values to model the unknown behavior of a defect. The X-fault model assumes an unknown (X) value at a potential defect location and allows error to propagate conservatively. A primary advantage of employing the X-fault model is to avoid the inherent problem of error masking and unmasking in multiple-defect diagnosis. However, the approaches described in [345–349] are limited to the scenarios when multiple

defects that affect a failing chip are logically/structurally close to each other.

Furthermore, prior work discussed up to this point and the methods presented in [290,330, 350–352] use various candidate-ranking heuristics to identify the correct candidates. However, such heuristics are explicitly developed based on intuition and/or domain knowledge, and thus are not guaranteed to work for every defect mechanism, design and/or process node. On the contrary, a candidate scoring procedure implicitly derived from test fail data can uncover the hidden correlations between a correct candidate and the observed circuit response, which otherwise could have been overlooked by manually constructed scoring models [274]. Thus, an alternative to rank candidates is machine learning.

Machine learning has been successfully applied in chip testing [353]. Specifically, in the area of diagnosis, machine learning has been used to optimize test data collection to make diagnosis more efficient [293–296, 296, 298], improve the accuracy and resolution of diagnosis itself [274, 299–304], and pinpoint yield-limiting layout geometries by analyzing a volume of diagnosis data [221, 225, 230–232, 305]. Prior work that deploys machine learning to boost the performance of diagnosis is surveyed in detail in Section 2.1. To the best of our knowledge, except our work of [274] that is presented in Chapter 2, machine learning has not been exploited during diagnosis for candidate prediction. (The technique presented in [306–308] analyzes a volume of diagnoses to predict a correct candidate, and thus complements MD-LearnX.)

3.2 Diagnosis Methodology

To address the drawbacks of prior work related to multiple-defect diagnosis discussed in Section 3.1, a single-chip diagnosis methodology called MD-LearnX is developed. Notable features of MD-LearnX include:

1. In addition to using a deterministic fault model (where the erroneous value is either

logic-0 or logic-1), the X-fault model is employed as well to prevent the elimination of a correct candidate. X-values stemming from more than one location in a design cannot obstruct the propagation of another error, thus circumventing one of the main drawbacks of multiple-defect diagnosis (that of error masking and unmasking).

2. As opposed to manually developing a complex candidate-scoring heuristic to identify the best candidate, MD-LearnX utilizes machine learning to create a scoring model that learns the latent correlations between a correct candidate and the tester response.

3. MD-LearnX uses design layout information to identify the physical defect type (e.g., interconnect open, interconnect bridge and cell internal defect) and behavior of each candidate. Back-end and front-end layout analysis techniques [146, 162] further strengthen MD-LearnX by improving its physical resolution.

4. MD-LearnX is applicable to defects exhibiting arbitrary misbehaviors. Unlike [321], it can effectively handle multiple byzantine defects without exhaustive enumeration of all fault combinations at a possible open defect location.

It should be noted that an approach that improves the quality of diagnosis by manipulating the test-set by either reordering, selecting or adding test patterns complement MD-LearnX [209, 210].

Figure 3.2 shows the overview of MD-LearnX. It is a three-phase diagnosis methodology. The sequence of analysis steps involved in each phase are marked with a different color in Figure 3.2. The first few steps common in each phase are discussed in Section 3.2.1. The first phase (Section 3.2.2) focuses on finding defects that mirror the behavior of well-established fault models. Such defects are henceforth referred to as class-1 defects. Phase 2 (Section 3.2.3) aims to identify a defect whose behavior deviates from that of each fault model considered in Phase 1 and whose error propagation path does not interfere with errors stemming from other defect locations for each failing pattern. Such defects are henceforth

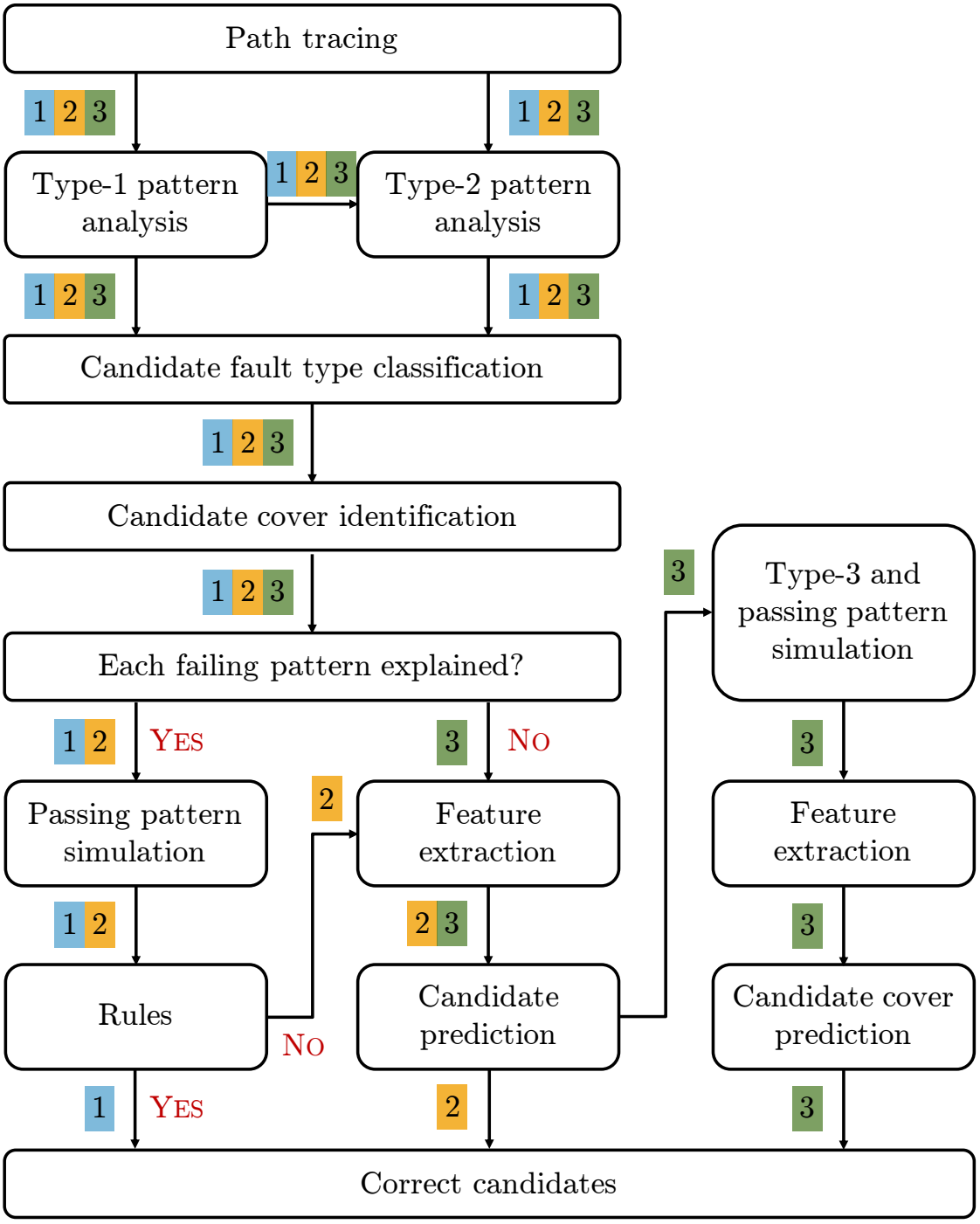


Figure 3.2: Overview of the proposed three-phase diagnosis methodology, MD-LearnX. The sequence of steps in each phase are marked with a different color.

referred to as class-2 defects. Finally, any failing chip left undiagnosed is analyzed in Phase 3 (Section 3.2.4). Such a failing chip is affected by multiple defects where errors manifesting from at least two defects interact with each other such that no single candidate can explain the observed response for at least one failing pattern.

Table 3.2 summarizes the class and multiplicity of defects targeted by each phase of MD-LearnX. The first and the second columns show the number of class-1 and class-2 defects. The third column shows whether the errors manifesting at more than one defect location interfere (i.e., block or assist) with each other. The fourth column shows the phase of MD-LearnX.

Each row in Table 3.2 represents the type of multiple defects that each phase endeavors to diagnose. Phase 1 aims to identify one or more class-1 defects (i.e., defect behaviors that echo classic fault models), where the propagation paths of errors stemming from more than one defect location are disjoint for each failing pattern. Phase 2 targets one or multiple, non-interacting defects, where the behavior of each defect deviates from classic fault models. If at least one class-1 defect and at least one class-2 defect reside in a failing chip, then Phase 1 and Phase 2 are employed if the errors originating at defect locations do not interact with each other for any failing pattern, and Phase 3 is employed if the errors interact for at least one failing pattern.

Class-1 defects	Class-2 defects	Error propagation paths	MD-LearnX phase
≥ 1	0		Phase 1
0	≥ 1		Phase 2
≥ 1	≥ 1	Disjoint for each failing pattern	Phase 1 and Phase 2
≥ 1	≥ 1	Overlapping for at least one failing pattern	Phase 3

Table 3.2: Class and multiplicity of defects targeted by each phase of MD-LearnX.

3.2.1 Phase 1-3 steps

MD-LearnX begins with path tracing [108–110], where it traces back from each failing output for each failing pattern to infer potential defective signals. Path tracing has been illustrated in Figure 2.1. Path tracing guarantees that it will find a defect location that aids in error propagation to at least one design output, even for multiple defects. It is however possible that an error stemming from a defect location only hinders error propagation from other defect locations to a design output. A defect location that cannot be sensitized for any failing pattern thus cannot be identified by path tracing; however, it is infeasible to locate such a defect besides explicitly considering all the defect locations in the design.

For each failing pattern, an X fault is simulated at each candidate location (one at a time) to identify faults that can explain[1] that pattern. Each such explained pattern is classified as a type-1 pattern. It should be noted that a type-1 pattern is defined in the literature with respect to a (temporary) stuck-at fault, not an X fault [136,137,320]. However, as discussed in Section 3.1, the X-fault model is not susceptible to error masking, and possibly averts the elimination of a candidate that represents an actual defect.

Each failing pattern is further analyzed to find a group of locations such that (a) their error propagation paths are disjoint, and (b) they can collectively explain that failing pattern. Each such pattern is classified as a type-2 pattern. This procedure is performed with respect to both stuck-at and X fault simulation. (A failing pattern is analyzed multiple times so that the final cover consisting of defect candidates resulting from the next couple of steps is optimal.) The remaining failing patterns, if any, are classified as type-3 patterns.

Next, physical defect locations associated with logical candidates that explain a type-1 or type-2 pattern are extracted from the design layout by examining their topology and the physical neighborhood. Each logical candidate is then mapped to a physical defect

[1] An X fault 'explains' a failing pattern when the set of simulated outputs that possess an X value subsume the erroneous circuit outputs.

candidate while keeping track of the set of failing patterns it explains. For example, if a logical candidate a explains pattern t_1, candidate b explains pattern t_2, and a and b form a physical bridge pair, then the defect candidate (a, b) explains both the patterns t_1 and t_2. Possible candidate fault types considered in this work are STUCK, CELL, BRIDGE and OPEN. The procedure of classifying a candidate into one of four fault types is further explained in Section 2.2.1.

The next step is to determine covers consisting of defect candidates via the set-cover approach such that (a) the size of a cover is minimum, i.e., a cover consists of a minimum number of defect candidates, and (b) defect candidates in a cover collectively explain each failing pattern that is either type-1 or type-2.

If each failing pattern is classified as either type-1 or type-2, that is, there are no type-3 patterns, it implies that the chip under diagnosis is affected by defects with disjoint error propagation paths for each failing pattern. The design (and the tester response) can essentially be partitioned such that each individual defect is diagnosed independently either in Phase 1 or Phase 2, as discussed in Sections 3.2.2 and 3.2.3, respectively.

3.2.2 Phase 1

As alluded to earlier, Phase 1 focuses on diagnosing defects that mimic the behavior of traditional fault models. Specifically, a set of strict rules are constructed for each candidate fault type to identify the correct cover of candidates. A defect candidate of a cover is deemed correct if it satisfies the rules (and hence represents a class-1 defect). The rules created for each fault type are identical to the ones created for LearnX (Section 2.2.1). The rules are more formally presented in Table 2.2, and described here briefly in Table 3.3.

Defect candidates of a cover that do not comply with the rules are further analyzed in Phase 2, which is especially geared towards the diagnosis of class-2 defects. It should be noted that the covers with the least number of class-2 defects are passed on to Phase 2 (by

Fault type	Rule
STUCK	Candidate passes for each passing pattern.
CELL	Cell passes consistency check [142–144, 171].
BRIDGE	Bridged nets have opposite polarities for failing patterns. Bridged nets have the same polarity for the passing patterns that sensitize one of the nets to a design output.
OPEN	Candidate passes for each passing pattern.
Common rule: Candidate explains each failing pattern. Here, a stuck-at fault is said to explain a failing pattern if the outputs that failed during simulation are identical to the observed failing outputs that are reachable from the fault location.	

Table 3.3: Rules for each candidate fault type in Phase 1 of MD-LearnX.

virtue of Occam's Razor).

3.2.3 Phase 2

A defect candidate, at this point of diagnosis, has the following properties; (a) it portrays behavior that cannot be modeled by the fault models employed in Phase 1, and (b) an error disseminating from the candidate location does not block or assist in error propagation of other possible defects in the circuit for any failing pattern. The primary objective of Phase 2 is to apply machine learning to discern the correct candidate for each defect.

Specifically, a frequently used supervised machine learning algorithm called a random forest [309] is used to classify a candidate as correct/incorrect. Forty-four features are extracted from the test data by comparing the test patterns and outputs observed on the tester and predicted by simulation. The features are derived from the stuck-at and the X fault simulation of a defect candidate, and likely, completely, represents its behavior. The designed features are the same as the ones used in LearnX (Chapter 2) and are listed in Tables 2.3 and 2.4.

The procedure of feature extraction is similar to Phase 2 of LearnX as well, albeit with

one notable difference. It should be recalled that Phase 2 of MD-LearnX identifies multiple defects whose fault effects do not intervene with each other. Thus, for each defect candidate, only the outputs that are reachable from its location are considered during feature extraction.

For each fault type (STUCK, CELL, BRIDGE and OPEN), a separate model is trained so that distinct characteristics relevant to each defect behavior can be learned. Training data is generated from diagnosing numerous virtual fail logs that are created by injecting and simulating single defects in the circuit. Hyperparameters of each model are optimized using cross-validation. Because only a single candidate can represent an actual defect, the training dataset is highly imbalanced. An optimum decision threshold for each trained model is thus selected using the Precision-Recall curve [274, 310]. Each defect candidate (according to its fault type) is then adjudged correct/incorrect by the corresponding trained ML model. The training procedure is similar to Phase 2 of LearnX and is explained in detail in Section 2.2.2.

Finally, a cover of defect candidates is said to represent actual defects when each defect candidate in the cover is deemed correct by either Phase 1 or Phase 2.

3.2.4 Phase 3

If there exists at least one type-3 pattern, it implies that the chip under diagnosis is affected by multiple defects with overlapping error propagation paths for at least one failing pattern.

In Phase 3, machine learning classification is applied at two different levels; first, at a defect level, where defect-type specific learning models built using the approach outlined in Phase 2 (Sections 2.2.2 and 3.2.3) classify each defect candidate as correct/incorrect, and, second, at a cover level, where the entire cover is predicted as correct/incorrect.

However, there are a few differences when defect-level classification is applied in Phase 3. First, a candidate cover is only simulated for type-1 and type-2 patterns up to this point, and not type-3 or passing patterns. Thus, among the 44 features presented in Table 2.3 and Table 2.4, 32 features that correspond to the failing patterns are used to train a defect-level

ML model in Phase 3. The features are shown in Table 3.4. Second, the training data is created from analyzing virtual fail logs that are produced by injecting and simulating multiple defects in the circuit. Third, in order to be conservative and avoid eliminating a correct candidate cover, a cover is analyzed further if at least one of its component defect candidates is predicted correct by the machine learning model built for the corresponding fault type. Each cover is then simulated for the remaining patterns (type-3 as well as passing patterns).

Next, machine learning is utilized at a candidate cover level to predict a cover of defect candidates as correct or incorrect. A single random forest is trained using the steps similar to that described in Section 2.2.2, with the difference being that each training instance is a candidate cover here. A cover is then said to represent actual defects when it is predicted correct by the machine learning model.

It should be noted that unlike [320, 326–329], multiple-fault simulation is performed here to extract features, and not explore the exponential search space. Unlike the work of [115, 131–144, 318, 319, 321–323], type-3 patterns are used here to further improve the quality of diagnosis.

3.3 Experiments

Two experiments are conducted to validate MD-LearnX: one is a defect injection and simulation experiment using four different designs (Section 3.3.2), and another that uses silicon failure data (Section 3.3.3). The diagnostic metrics that are employed to evaluate the performance of MD-LearnX are discussed in Section 3.3.1. The effectiveness of MD-LearnX is compared with leading-edge commercial diagnosis in the experiments. In each experiment, the flow illustrated in Figure 3.2 is used to diagnose a fail log. Each fail log is also examined by commercial diagnosis, where only the top-scoring candidate for each suspected defect is

Feature	Feature description
$\|TFSF_p\|/\|TF^p\|$	Ratio of the number of TFSF patterns to the number of TF patterns
$\|TFSF_p\|/\|SF^p\|$	Ratio of the number of TFSF patterns to the number of SF patterns
$\|TFSP_p\|/\|TF^p\|$	Ratio of the number of TFSP patterns to the number of TF patterns
$\|TFSP_p\|/\|SP^p\|$	Ratio of the number of TFSP patterns to the number of SP patterns
$\sum_{p\in TFSF_p} \|TFSF_o^p\|/\sum_{p\in P} \|TF_o^p\|$	Ratio of the number of TFSF outputs for TFSF patterns to the number of TF outputs
$\sum_{p\in TFSF_p} \|TFSF_o^p\|/\sum_{p\in P} \|SF_o^p\|$	Ratio of the number of TFSF outputs for TFSF patterns to the number of SF outputs
$\sum_{p\in TFSF_p} \|TPSF_o^p\|/\sum_{p\in P} \|TP_o^p\|$	Ratio of the number of TPSF outputs for TFSF patterns to the number of TP outputs
$\sum_{p\in TFSF_p} \|TPSF_o^p\|/\sum_{p\in P} \|SF_o^p\|$	Ratio of the number of TPSF outputs for TFSF patterns to the number of SF outputs
$\sum_{p\in TFSF_p} \|TFSP_o^p\|/\sum_{p\in P} \|TF_o^p\|$	Ratio of the number of TFSP outputs for TFSF patterns to the number of TF outputs
$\sum_{p\in TFSF_p} \|TFSP_o^p\|/\sum_{p\in P} \|SP_o^p\|$	Ratio of the number of TFSP outputs for TFSF patterns to the number of SP outputs
$\sum_{p\in TFSF_p} \|TPSP_o^p\|/\sum_{p\in P} \|TP_o^p\|$	Ratio of the number of TPSP outputs for TFSF patterns to the number of TP outputs
$\sum_{p\in TFSF_p} \|TPSP_o^p\|/\sum_{p\in P} \|SP_o^p\|$	Ratio of the number of TPSP outputs for TFSF patterns to the number of SP outputs
$\sum_{p\in TFSP_p} \|TFSP_o^p\|/\sum_{p\in P} \|TF_o^p\|$	Ratio of the number of TFSP outputs for TFSP patterns to the number of TF outputs
$\sum_{p\in TFSP_p} \|TFSP_o^p\|/\sum_{p\in P} \|SP_o^p\|$	Ratio of the number of TFSP outputs for TFSP patterns to the number of SP outputs
$\sum_{p\in TFSP_p} \|TPSP_o^p\|/\sum_{p\in P} \|TP_o^p\|$	Ratio of the number of TPSP outputs for TFSP patterns to the number of TP outputs
$\sum_{p\in TFSP_p} \|TPSP_o^p\|/\sum_{p\in P} \|SP_o^p\|$	Ratio of the number of TPSP outputs for TFSP patterns to the number of SP outputs

Table 3.4: Features extracted from the test data for Phase 3. Sixteen features for each type of simulation are identified, resulting in a total of 32 pattern-level features.

considered at its output.

3.3.1 Diagnostic Metrics

Ideally, a diagnosis methodology should correctly report a single candidate and deduce the precise logic behavior for each defect in a failing chip. MD-LearnX concentrates on the former, i.e., accurate defect localization. The latter, i.e., defect characterization in terms of its behavior, is the focus of Chapters 4 and 5.

The following metrics are used to compare the efficacy of MD-LearnX with commercial diagnosis.

- Diagnosability [341–343]: Diagnosability is defined as the ratio of the number of defect locations that are correctly identified to the number of defects in a (virtual) failing chip. Thus, the ideal value of diagnosability is one, when each defect is accurately located by diagnosis.

- Precision: Precision is defined as the ratio of the number of defect locations that are correctly identified to the number of defect locations returned by diagnosis. Higher the precision, better is the quality of diagnosis.

- *Home run*: The definition of *home run* is adapted here from its description in Section 2.3.1 to accommodate multiple defects. Diagnosis is said to hit a home run when diagnosis is perfect, i.e., when a single correct candidate is reported for each defect affecting a failing chip. Mathematically, *home run* is equal to one when diagnosability and precision are simultaneously equal to one, and zero otherwise.

3.3.2 Simulation Experiment

A simulation-based experiment is performed for evaluating MD-LearnX using the four designs analyzed in Section 2.3.2, namely, an AES core (or simply, "AES"), an IWLS'05 benchmark

referred to as "DES", the L2 cache of the OpenSPARC T2 processor called "L2B", and an ITC'99 benchmark circuit termed as "B18". Realistic physical defect behaviors associated with bridge, open and cell defects (including byzantine bridges [72] and opens [147]) are utilized.

For each design and defect multiplicity, 1,000 virtual fail logs are generated by uniquely and randomly selecting the location and the behavior of each injected defect. For each design, an additional 1,000 fail logs are used to produce the training dataset (for Phase 3), while ensuring that each multiple fault injected is different. Each fail log is diagnosed using MD-LearnX and two state-of-the-art commercial diagnosis tools. The criteria established in Section 3.3.1 are used to evaluate each diagnosis technique.

The rest of this section is organized as follows. Section 3.3.2.1 compares the quality of diagnosis stemming from MD-LearnX and commercial diagnosis for each of the four designs analyzed. Section 3.3.2.2 summarizes the performance of MD-LearnX over all the designs.

3.3.2.1 Results

Figures 3.3 through 3.6 show the probability density distribution of diagnosability achieved by MD-LearnX and commercial diagnosis for different designs. A density plot can be thought of as a smoothed histogram that is symmetrical about its axis, and displays the density of the distribution using the width of the plot. Thus, a wider distribution at a particular value indicates more density at that value. A density plot is preferred over a box plot because while the latter only displays the quartiles of the distribution, the former depicts the entire distribution.

In each figure, the horizontal axis represents the injected defect multiplicity and the vertical axis shows the distribution of diagnosability. Each figure contains three density plots for each defect multiplicity, one for each diagnosis technique. A solid black line across each plot indicates the mean of the distribution of diagnosability.

It is evident from the shapes of each plot in Figures 3.3-3.6 that MD-LearnX is more effective than commercial diagnosis. Observations specific to each design are as follows.

1. **AES**: Figure 3.3 reveals that the average diagnosability of MD-LearnX is 0.77, which is an improvement of 17.7% over Tool 1 and 2X times Tool 2. On average, MD-LearnX diagnoses more than half (52.9%) of the fail logs with a diagnosability of one, which is 20.9% more than Tool 1 and 3X times Tool 2. For multiple defects (i.e., when defect multiplicity is more than 1), the diagnosability of MD-LearnX decreases to 0.73, but its relative effectiveness increases. Specifically, the diagnosability of MD-LearnX is 20.4% and 2.3X times better than Tool 1 and Tool 2, respectively, for multiple defects. MD-LearnX correctly locates each injected defect in 46.4% of fail logs when the injected defect multiplicity is at least two, which is 26.7% more than Tool 1 and 5X times Tool 2. Although MD-LearnX is unable to correctly identify any defect for 7.9% of fail logs, commercial diagnosis is inadequate more often; specifically, Tool 1 and Tool 2 report a diagnosability of zero for 18.8% and 5.4X more fail logs, respectively.

2. **DES**: It is observed from Figure 3.4 that the diagnosability of MD-LearnX is 0.60, on average, while the average diagnosability of Tool 1 (Tool 2) is 0.55 (0.42). MD-LearnX achieves an ideal diagnosability for 31.8% of fail logs, 3.3% (50%) more fail logs than Tool 1 (Tool 2). When the failing circuit is affected by five or more defects, MD-LearnX performs even better. Specifically, the improvement in ideal diagnosability over Tool 1 is 19.0% and over Tool 2 is 96.5%. Moreover, Tool 1 (Tool 2) cannot locate any defect correctly for 12.3% (40.5%) of fail logs, which is 30.0% more than (4.3X times) MD-LearnX.

3. **L2B**: Figure 3.5 shows that MD-LearnX attains an average diagnosability of 0.63, which is an improvement of 25.9% over Tool 1 and 2.7X times Tool 2. MD-LearnX returns a diagnosability of one for 44.7% of fail logs, which is 2.2X (2.7X) times more

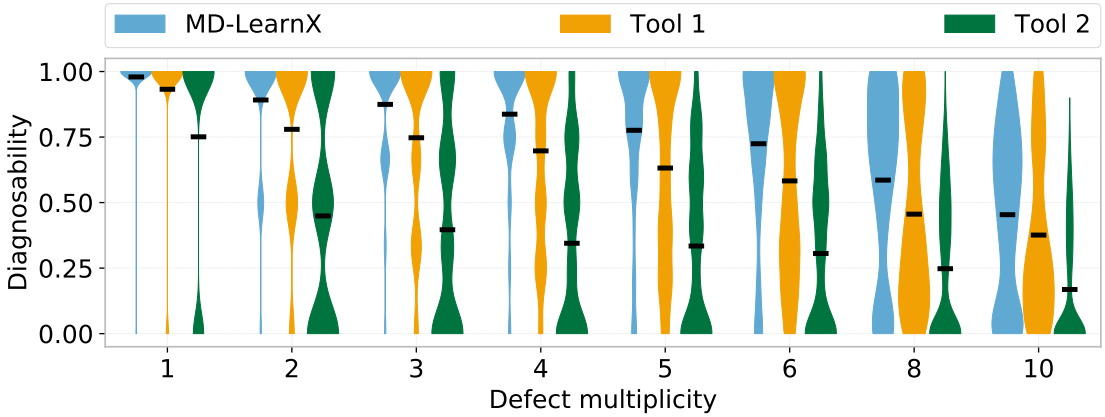


Figure 3.3: Probability density distribution of diagnosability for the design "AES".

often than Tool 1 (Tool 2). The effectiveness of MD-LearnX in diagnosing at least five defects is noteworthy – on average, the improvement in diagnosability over Tool 1 (Tool 2) is 38.9% (11.4X), and MD-LearnX correctly identifies each injected defect 38X (120X) times more often than Tool 1 (Tool 2).

4. **B18**: It is seen from Figure 3.6 that the average diagnosability of MD-LearnX is 0.66, and is 0.5 and 0.29 for Tool 1 and Tool 2, respectively. MD-LearnX achieves a diagnosability of one for 45.0% of fail logs, which is 2X (2.6X) times Tool 1 (Tool 2). For multiple defects, the improvement in average diagnosability over Tool 1 (Tool 2) increases to 35.8% (2.7X). Additionally, MD-LearnX returns ideal diagnosability 2.8X (4.2X) more often than Tool 1 (Tool 2) for multiple defects. MD-LearnX is quite effective when defect multiplicity is equal to or more than five; it accurately locates each injected defect for 28.2% of fail logs, which is 36X (28X) times Tool 1 (Tool 2).

Figures 3.7-3.10 compare the probability density distribution of precision attained by MD-LearnX with commercial diagnosis for different designs. The difference in the shapes of each plot in each figure illustrates the superior performance of LearnX over commercial diagnosis. Observations specific to each design are as follows.

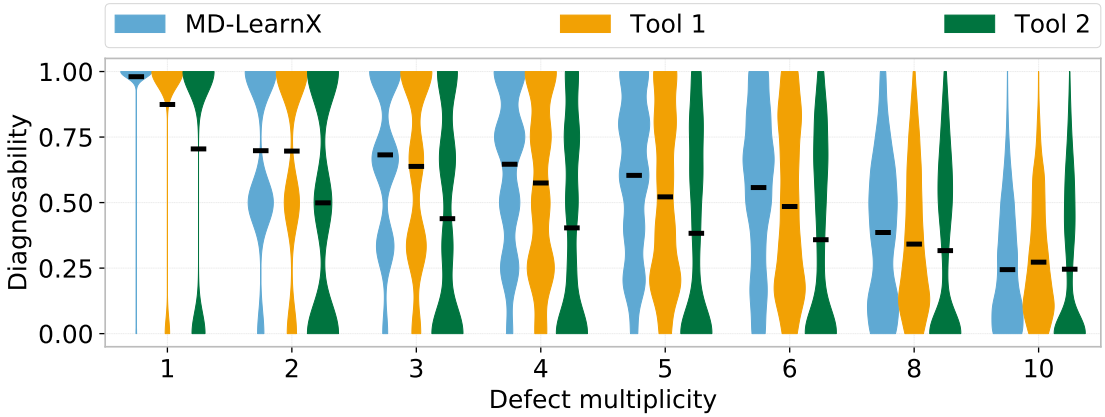


Figure 3.4: Probability density distribution of diagnosability for the design "DES".

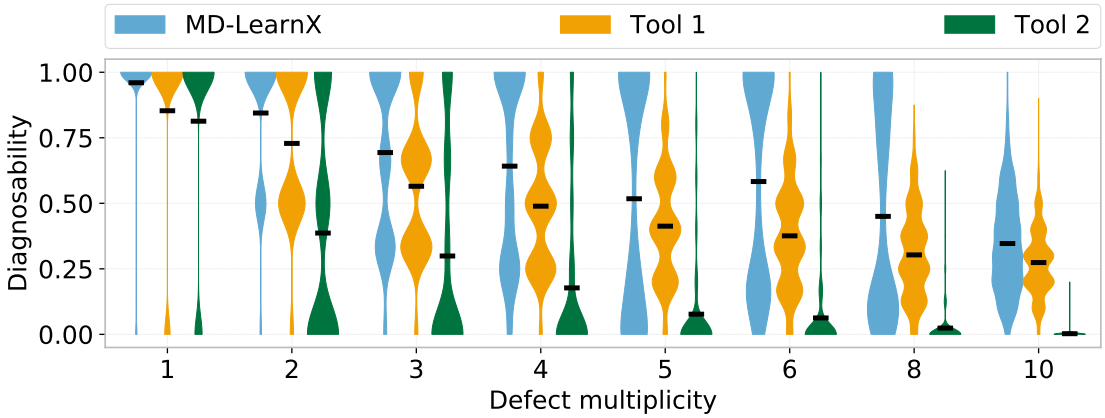


Figure 3.5: Probability density distribution of diagnosability for the design "L2B".

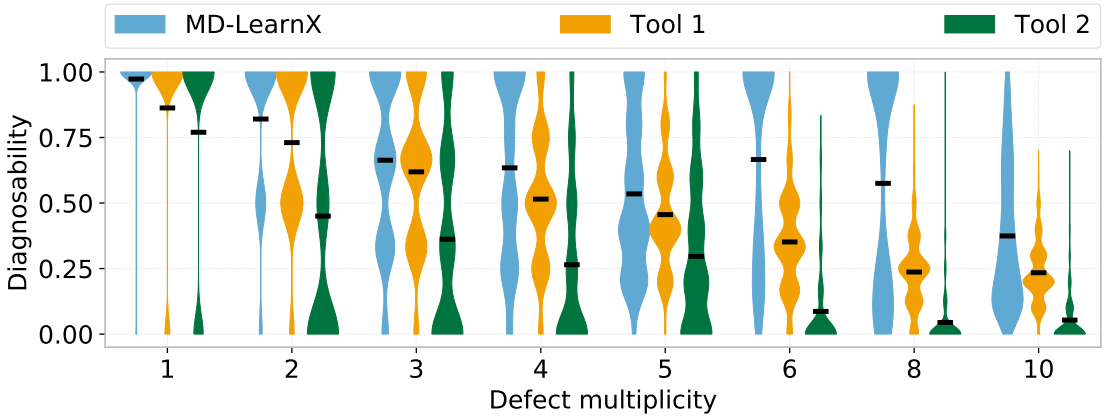


Figure 3.6: Probability density distribution of diagnosability for the design "B18".

1. **AES**: Figure 3.7 reveals that the average precision of MD-LearnX is 0.66, which is an improvement of 95.1% over Tool 1 and 2.2X times Tool 2. On average, MD-LearnX diagnoses 32.3% of fail logs with a precision of one, which is 2.4X times both the tools. For multiple defects (i.e., when defect multiplicity > 1), the precision of MD-LearnX is 2.1X (2.5X) times Tool 1 (Tool 2). MD-LearnX does not report an incorrect candidate for 27.3% of fail logs when the injected defect multiplicity is at least two, which is 5.2X (5.6X) times more often than Tool 1 (Tool 2).

2. **DES**: It is observed from Figure 3.8 that the precision of MD-LearnX is 0.56, on average, while the average precision of Tool 1 (Tool 2) is 0.28 (0.24). MD-LearnX achieves an ideal precision (i.e., when precision is equal to one) for 25.9% of fail logs, which is 3.8X (3.3X) times Tool 1 (Tool 2). When defect multiplicity is at least two, each candidate reported by MD-LearnX is correct for 21.2% of fail logs, while the number of such fail logs is only 3.3% (2.7%) for Tool 1 (Tool 2). When a failing circuit is affected by five or more defects, MD-LearnX performs even better. Specifically, the improvement in ideal precision over Tool 1 is 5.7X and over Tool 2 is 17X.

3. **L2B**: Figure 3.9 shows that MD-LearnX attains an average precision of 0.74, which is an improvement of 45.0% over Tool 1 and is 4.4X times Tool 2. MD-LearnX returns a precision of one for 57.8% of fail logs, which is 82.2% more than Tool 1 and 6.8X times Tool 2. When defect multiplicity is at least five, the average precision of MD-LearnX is 41.6% more than Tool 1 and 9.4X times Tool 2; the ideal precision is 70.5% more than Tool 1 and 11X times Tool 2.

4. **B18**: It is seen from Figure 3.10 that the average precision of MD-LearnX is 0.61, which is 57.6% and 2.3X times better than Tool 1 and Tool 2, respectively. MD-LearnX achieves a precision of one for 36.1% of fail logs, which is 2.4X (2.9X) times Tool 1 (Tool 2). For multiple defects, the improvement in average precision over Tool

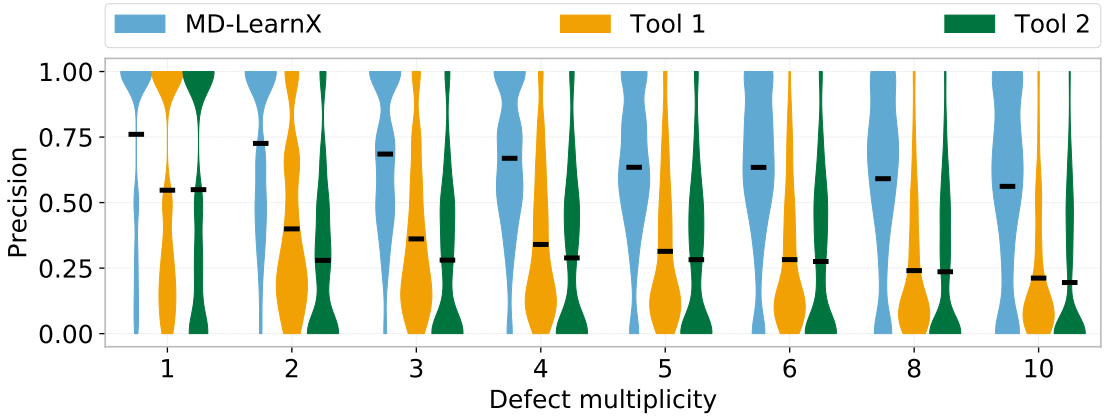


Figure 3.7: Probability density distribution of precision for the design "AES".

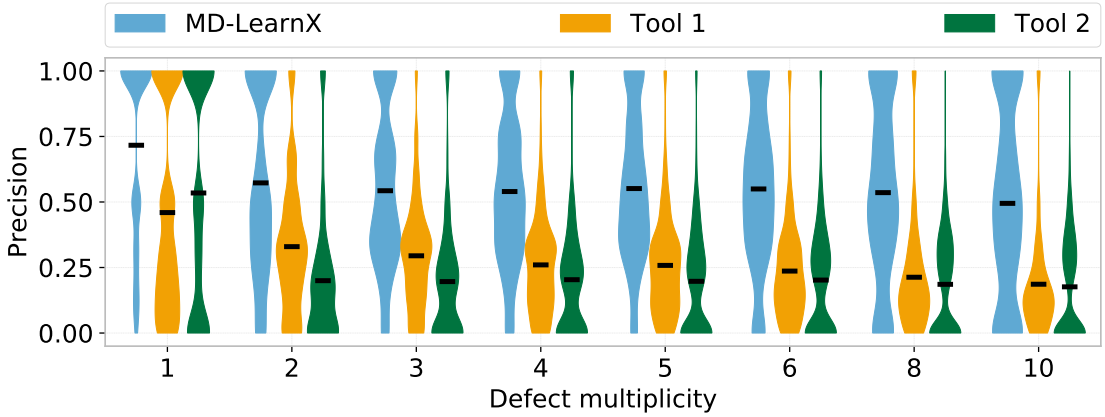


Figure 3.8: Probability density distribution of precision for the design "DES".

1 (Tool 2) increases to 60.5% (2.4X). Additionally, MD-LearnX returns ideal precision 2.6X (3.1X) more often than Tool 1 (Tool 2) for multiple defects.

Next, the capability of LearnX to accomplish a perfect diagnosis, i.e., when each defect residing in a failing chip is correctly identified with a single candidate, is investigated. Figure 3.11 reports the number of home runs hit by MD-LearnX for different defect multiplicities for the four designs analyzed. Figure 3.11 consists of four plots, one for each design. It is seen from Figure 3.11 that MD-LearnX achieves impressive improvement over commercial diagnosis. Figure 3.11 reveals the following for each design.

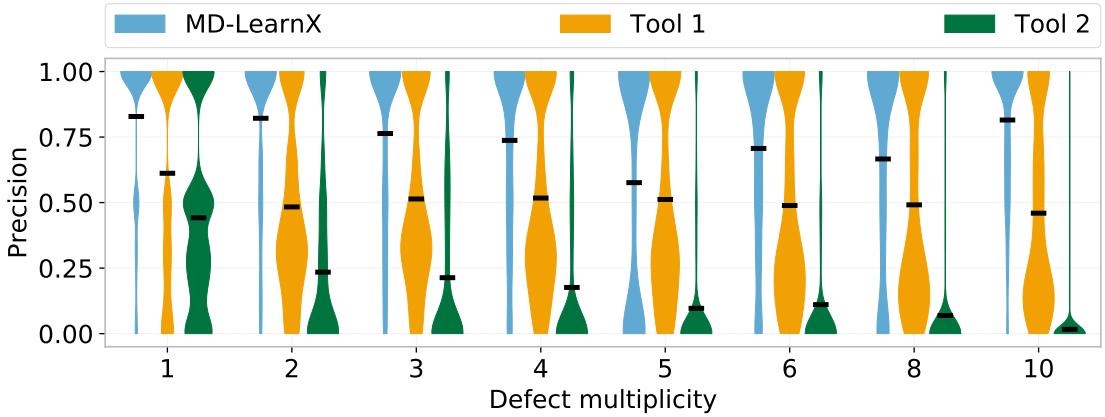


Figure 3.9: Probability density distribution of precision for the design "L2B".

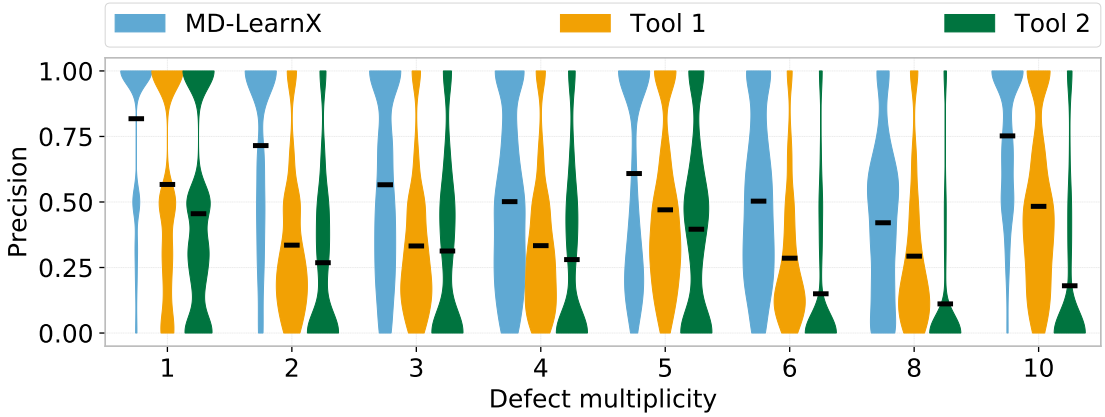


Figure 3.10: Probability density distribution of precision for the design "B18".

1. **AES**: It is observed from Figure 3.11(a) that MD-LearnX delivers a home run for 24.2% of fail logs, on average, which is 2.7X (3.7X) times Tool 1 (Tool 2). When defect multiplicity is greater than or equal to five, the number of perfect diagnoses by MD-LearnX reduces to 6.6%, but reports a perfect diagnosis 6.7X (130X) more often than Tool 1 (Tool 2).

2. **DES**: Figure 3.11(b) shows that MD-LearnX hits a home run for 14.6% of fail logs, which is 3.2X times Tool 1 and 2.6X times Tool 2. For multiple defects, MD-LearnX hits a home run for 8.4% of fail logs, while commercial diagnosis hits for less than 1.0%.

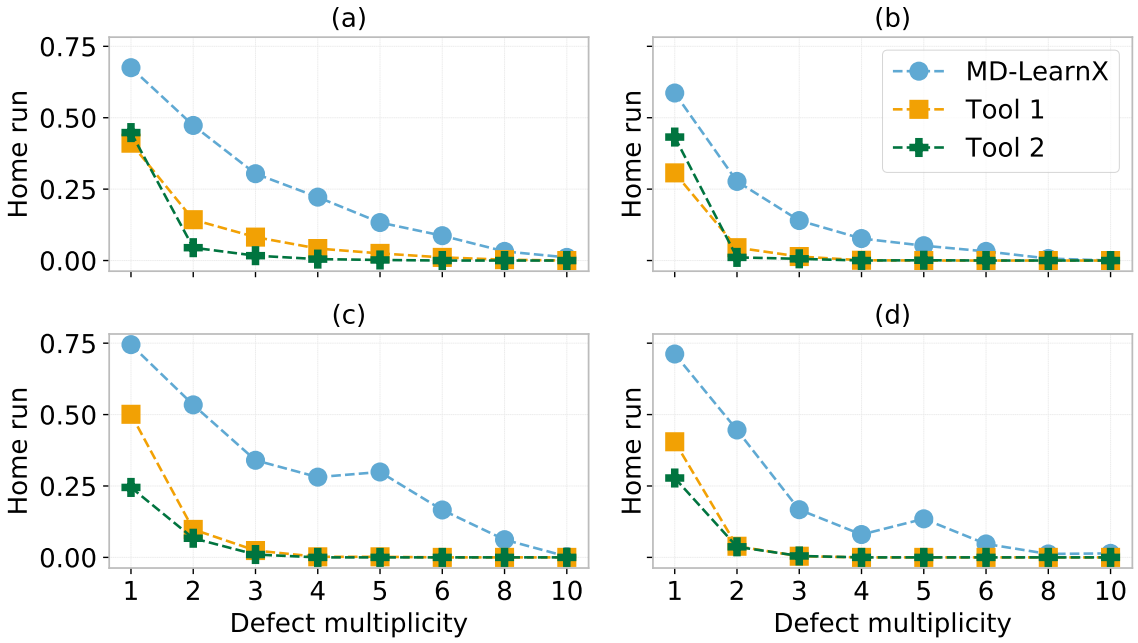


Figure 3.11: *Home run* of MD-LearnX compared with commercial diagnosis for (a) AES, (b) DES, (c) L2B, and (d) B18.

When defect multiplicity is at least five, while commercial diagnosis virtually does not return any perfect diagnosis, MD-LearnX delivers a home run for 2.3% of fail logs.

3. **L2B**: It is seen from Figure 3.11(c) that the number of perfect diagnoses returned by MD-LearnX is 30.4%, and by Tool 1 (Tool 2) is 7.8% (4.0%). When defect multiplicity is greater than or equal to five, while commercial diagnosis virtually does not return any perfect diagnosis, MD-LearnX hits a home run for 13.3% of fail logs.

4. **B18**: Figure 3.11(d) reveals that MD-LearnX delivers a home run for 20.2% of fail logs, which is 3.6X (5X) times Tool 1 (Tool 2). For defect multiplicity greater than one, the number of perfect diagnoses returned by MD-LearnX is 20 times commercial diagnosis. When the number of injected defects is at least five, MD-LearnX hits a home run for 5.2% of fail logs, and commercial diagnosis does not hit a single home run.

Furthermore, MD-LearnX is evaluated based on its ability to estimate the defect multiplicity. Figure 3.12 shows the probability density distribution of the number of defects estimated (y-axis) by MD-LearnX and commercial diagnosis for each injected defect multiplicity (x-axis). A solid black line across each plot indicates the median of the distribution of estimated defect multiplicity.

It can be seen from Figure 3.12 that the median estimated defect multiplicity is close to the ideal value for smaller values of injected defect multiplicity. MD-LearnX correctly predicts the number of defects for 62.4% of fail logs when the actual defect multiplicity is at most four, which is an enhancement of 18.4% over Tool 1 and 77.8% over Tool 2.

Figure 3.12 expectedly, however, reveals that the likelihood of misprediction increases for larger values of injected defects. Specifically, when the number of injected defects is at least five, MD-LearnX accurately estimates the defect multiplicity for 17.8% of fail logs, compared to 15.3% by Tool 1 and 6.6% by Tool 2. Moreover, the number of correct estimations by

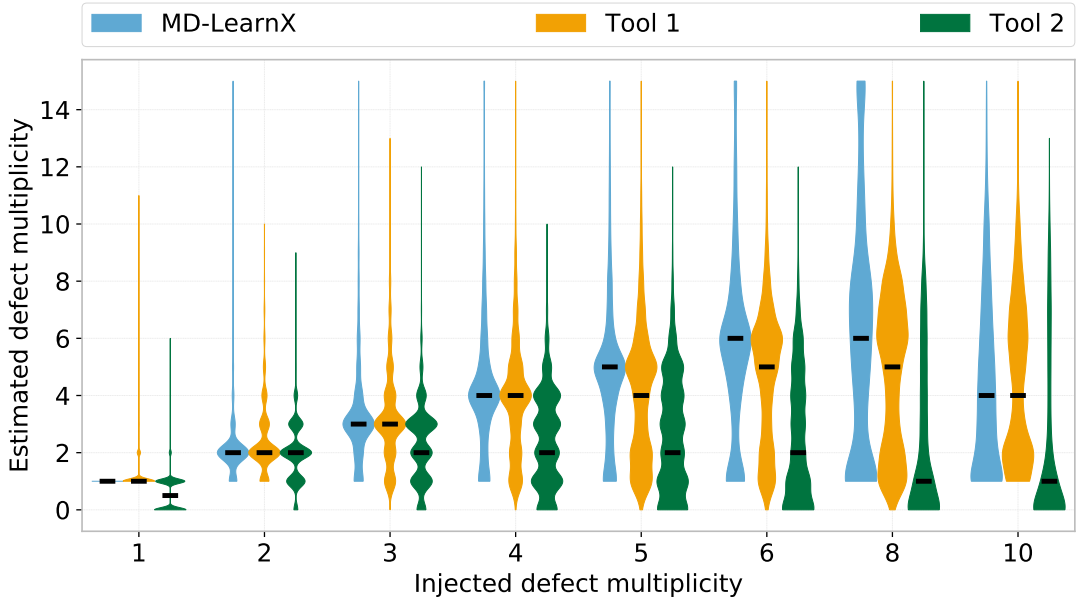


Figure 3.12: Distribution of defect multiplicity estimated by MD-LearnX and commercial diagnosis.

MD-LearnX decreases to 3.2%, when the actual defect multiplicity is 10, which is 25.0% more than and 2.3X times Tool 1 and Tool 2, respectively.

On average, MD-LearnX correctly determines the defect multiplicity for 40.1% fail logs, which is 17.6% (91.8%) more than Tool 1 (Tool 2).

When the runtime of MD-LearnX is compared with commercial diagnosis, it is observed that, on average, MD-LearnX is 25.2% slower than Tool 1 and 59.7% faster than Tool 2. This comparison does not include the one-time cost (per design) of creating the machine learning models for phases 2 and 3. For instance, for the largest design, AES, it takes 5.7 hours to learn the models. However, the primary goal here is to show the effectiveness (in terms of the diagnostic metrics discussed in Section 3.3.1) rather than the runtime efficiency of MD-LearnX. One of the future goals is to optimize the runtime of MD-LearnX.

3.3.2.2 Results Summary

Figure 3.13 illustrates the average diagnosability (i.e., the proportion of defects located correctly) attained by MD-LearnX for the four designs examined, irrespective of the number of injected defects, and compares it with commercial diagnosis. The x-axis shows the diagnosability for each design and for each diagnosis approach. The y-axis represents each design. The values/percentages displayed besides the plot-bars for Tool 1 and Tool 2 emphasize the improvement in diagnosability attained by MD-LearnX.

It can be seen from Figure 3.13 that the maximum improvement by MD-LearnX over Tool 1 is observed for B18 (specifically, 21.5%), and over Tool 2 for L2B (specifically, 66.5%). The diagnosability of MD-LearnX is at least 0.75 (for DES) and at most 0.85 (for AES). On the other hand, the diagnosability ranges from 0.64 to 0.76 for Tool 1, and from 0.45 to 0.52 for Tool 2.

Figure 3.14 reports the average precision (i.e., the proportion of defect candidates that are correctly identified to the number of candidates suspected) achieved by MD-LearnX

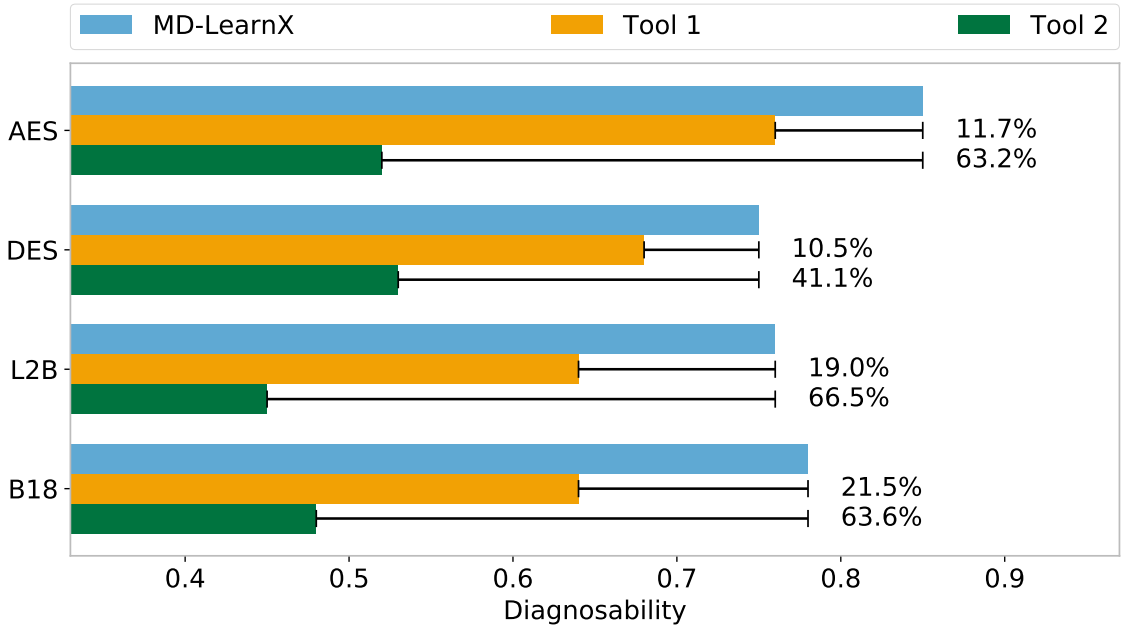


Figure 3.13: Diagnosability achieved by MD-LearnX and commercial diagnosis.

and commercial diagnosis for the four designs examined. The precision attained by LearnX varies from 0.62 to 0.77. The enhancement in precision attained by MD-LearnX over Tool 1 (Tool 2) is maximum for DES (L2B). Specifically, each candidate reported by MD-LearnX is correct for up to 78.3% more fail logs than Tool 1, and 2.8X times Tool 2.

Figure 3.15 accentuates the number of perfect diagnoses (i.e., when each defect residing in a failing chip is correctly identified with a single candidate) reported by MD-LearnX and commercial diagnosis for the four designs examined. The *home run* attained by MD-LearnX varies from 0.32 to 0.47. MD-LearnX returns a perfect diagnosis for at least 91.8% more fail logs than and at most 2.2X times Tool 1; it hits a home run for at least 57.1% more fail logs and at most 4X more often than Tool 2.

Results from a simulation-based experiment presented in Section 3.3.2 and summarized in this section demonstrate the superior performance of MD-LearnX over state-of-the-art commercial diagnosis. High diagnosability, precision and *home run* imply that there is

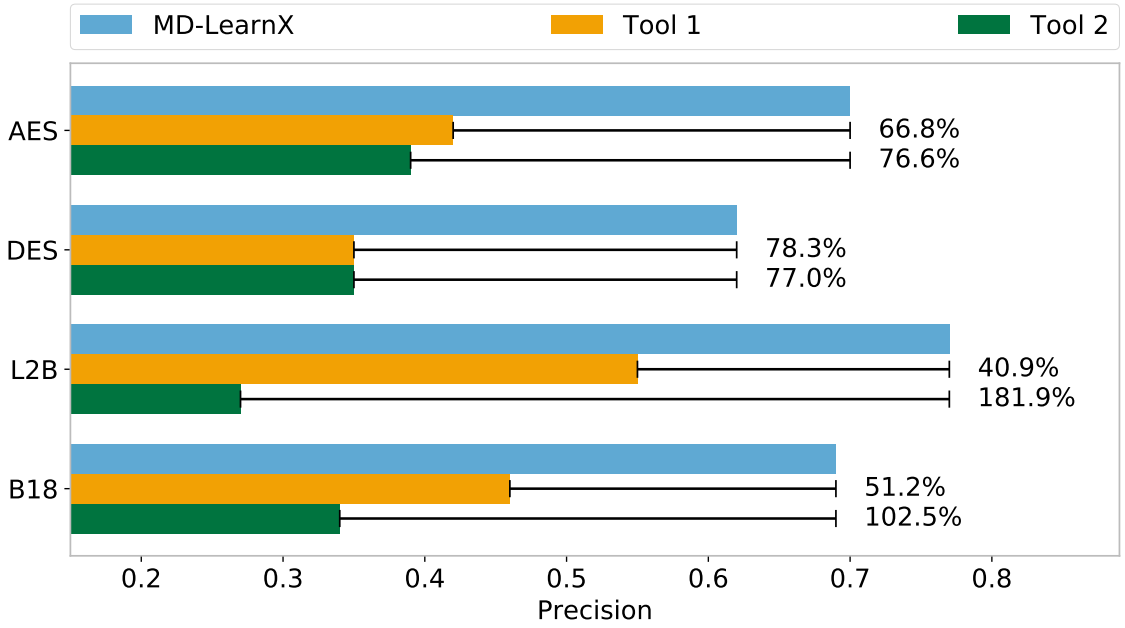


Figure 3.14: Precision achieved by MD-LearnX and commercial diagnosis.

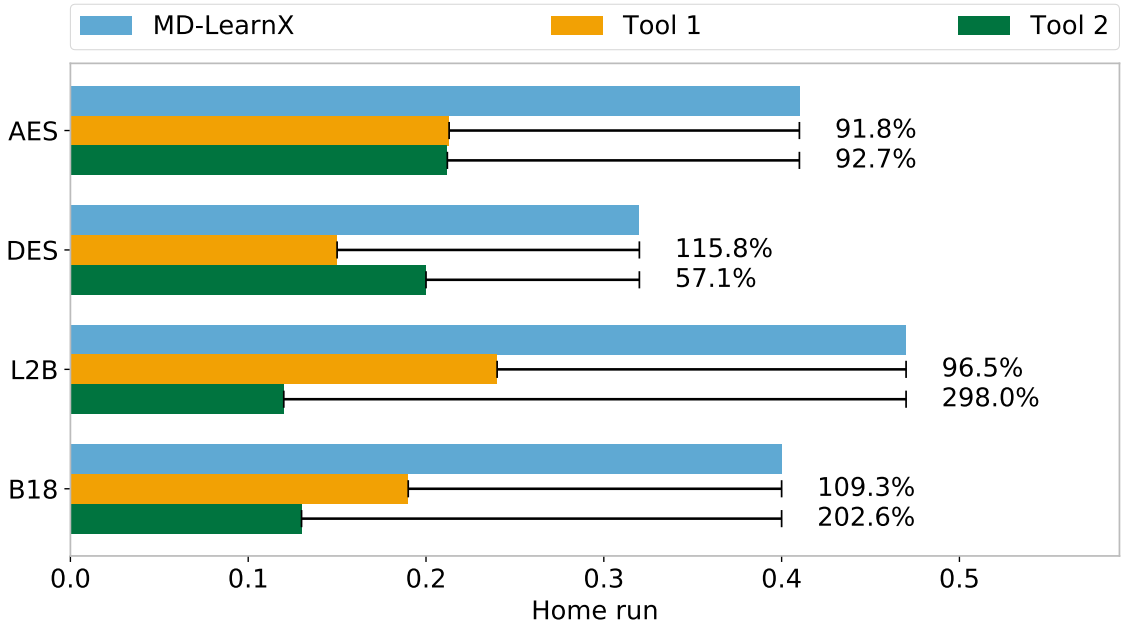


Figure 3.15: *Home run* achieved by MD-LearnX and commercial diagnosis.

less ambiguity of whether a candidate represents a defect. Consequently, a more precise distribution of likely yield detractors can be derived, which could lead to higher PFA success rate, and possibly, accelerated yield learning.

3.3.3 Silicon Experiment

The diagnosis potential of MD-LearnX is further substantiated by using an industrial design fabricated in an advanced process node. The setup for the conducted experiment is described in Section 3.3.3.1. It conveys basic information about the design and the silicon failure data that is available to study the effectiveness of MD-LearnX. The results of the experiment are discussed in Section 3.3.3.2 to explore how MD-LearnX performs in the real world.

3.3.3.1 Setup

MD-LearnX is applied to actual failing chips to evaluate its performance. The silicon failure data comes from the same design that is used to evaluate LearnX in Section 2.3.3. The MD-LearnX flow illustrated in Figure 3.2 is applied to each fail log. Each fail log is also analyzed by commercial diagnosis, where only the top-scoring candidates for each implicated defect are considered as its output. Note that the available failure data complies to Tool 1 and hence, the diagnosis quality attained by MD-LearnX is compared with Tool 1 only.

3.3.3.2 Results

The number of defect candidates reported by MD-LearnX are compared with commercial diagnosis for 36 failing chips that have been PFA'ed. For each of the 36 fail logs, it is seen that MD-LearnX (and commercial diagnosis) correctly localizes the defects that are PFA'ed. **More importantly, MD-LearnX returns fewer defect candidates than commercial diagnosis, on average, without sacrificing accuracy.** Figure 3.16 compares the number

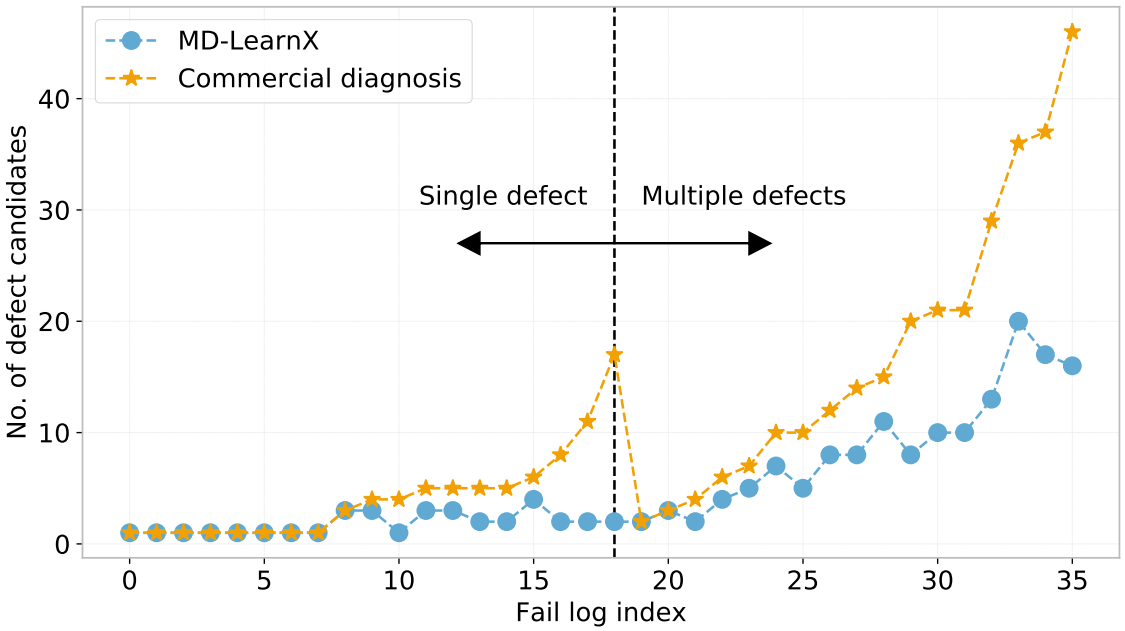


Figure 3.16: Number of candidates returned by MD-LearnX and commercial diagnosis for 36 failing chips that are PFA'ed.

of defect candidates (y-axis) returned by MD-LearnX and commercial diagnosis for each fail log (x-axis) for which the PFA results are available.

Figure 3.16 reveals that MD-LearnX returns fewer defect candidates than commercial diagnosis for 25 (69.4%) fail logs, while returning the same candidates for other fail logs, without losing accuracy. On average, 5.3 fewer candidates per fail log are returned, with maximum improvement being 88.2%. Commercial diagnosis reports a maximum of 46 candidates for a fail log, while MD-LearnX returns a maximum of 20 candidates.

Furthermore, for the 17 fail logs diagnosed with multiple defects, the improvement is more profound. Specifically, 8.5 fewer candidates per fail log are returned, on average. In addition, MD-LearnX reports fewer candidates for 15 (88.2%) fail logs.

3.4 Conclusion

In this chapter, a single-chip diagnosis methodology called MD-LearnX is described to effectively diagnose multiple defects. It is a physically-aware, three-phase diagnosis methodology. Phases 1 and 2 focus on diagnosing a chip affected by a single defect, or multiple defects that do not interact with each other. While Phase 1 centers on finding a defect that echoes the behavior of classic fault models via a set of deterministic rules, Phase 2 concentrates on identifying a defect through machine learning. A scoring model (separate for each fault type) that learns the hidden correlations between the tester response and the correct candidate is created to identify the correct candidate.

Phase 3, on the other hand, is adept in diagnosing a chip affected with multiple interacting defects. First, similar to Phase 2, it applies machine learning at a defect candidate level using failing patterns that can be explained by multiple, non-interacting defect candidates. Then, it employs machine learning at a candidate cover level using the passing and the remaining failing patterns to identify the cover of defect candidates that corresponds to actual defects.

A comprehensive simulation-based experiment is conducted to assess MD-LearnX, where a total of 28,000 faulty circuits with varying defect multiplicities and behaviors are created and analyzed. Three metrics, namely, diagnosability (i.e., proportion of injected defects that are correctly located), precision (i.e., proportion of reported defects that are correctly identified) and *home run* (i.e., when a single correct candidate is reported for each injected defect), are employed to measure the performance of MD-LearnX. The effectiveness of MD-LearnX is also compared with two leading-edge commercial diagnosis tools (that are referred to as Tool 1 and Tool 2).

The proposed methodology achieves an average diagnosability of 0.78, which is 15.4% and 58.2% better than Tool 1 and Tool 2, respectively. The average precision for MD-LearnX is 0.7, and is an improvement of 57.0% (2X) over Tool 1 (Tool 2). Additionally, MD-LearnX

hits a home run for 40.0% of fail logs, which is twice as often as commercial diagnosis.

The efficacy of MD-LearnX is impressive for large values of defect multiplicity. Specifically, when the number of injected defects is at least five, the diagnosability of MD-LearnX is 22.8% higher than Tool 1 and 2.3X times Tool 2, on average. MD-LearnX returns a correct candidate for each reported defect 2.4X (6.2X) more often than Tool 1 (Tool 2). Moreover, MD-LearnX delivers a home run for 6.8% of fail logs; however, commercial diagnosis returns a perfect diagnosis for less than 0.3% of fail logs.

The capability of MD-LearnX is further demonstrated with a silicon experiment, where 36 fail logs whose PFA results are available are diagnosed. It is seen that MD-LearnX returns fewer candidates than commercial diagnosis for 69.4% of the fail logs, without sacrificing accuracy. Moreover, MD-LearnX reports 5.3 fewer candidates per fail log, on average.

Software diagnosis, which is the first step in failure analysis, is the backbone of yield learning and monitoring. High diagnosis quality can effectively guide and accelerate PFA, likely facilitating yield ramp. This chapter, in particular, focuses on the localization of a defect in a failing chip. The next chapter, Chapter 4, describes a methodology that directs its efforts in improving the physical resolution as well as deriving the precise logic behavior for a back-end defect.

Chapter 4

PADLOC: Physically-Aware Defect Localization and Characterization

Chapter 2 introduces a diagnosis methodology called LearnX that focuses on identifying the location of a defect affecting a failing chip. MD-LearnX, that is presented in Chapter 3, builds on LearnX to effectively localize multiple defects in a failing chip. In addition to localizing a defect, LearnX and MD-LearnX identify the defect type of a candidate as well. However, as detailed in Chapter 1, the ultimate goal of diagnosis should be to logically characterize a defect. In other words, a diagnosis methodology should derive the precise logic behavior of the defect (in contrast to just finding the defect type) in addition to pinpointing its physical location. Furthermore, LearnX and MD-LearnX use design layout information to identify physically-feasible candidates in the logic netlist but do not endeavor to localize a defect beyond that.

To logically characterize a defect in terms of its behavior and further improve its physical location (in terms of its $x - y - z$ location in the layout), a methodology called PADLOC (Physically-Aware Defect LOcalization and Characterization) is developed and is the focus of this chapter. PADLOC further physically localizes a defect identified by LearnX/MD-

LearnX by finding the subnets that are more likely to correspond to the actual defect. Additionally, it logically characterizes a defect by deriving its behavior based on the activity of its neighborhood (i.e., the nets that are physically close and logically related). PADLOC is applicable to back-end defects; front-end (i.e., intra-cell) defects are the focus of Chapter 5.

The rest of the chapter is organized as follows. Section 4.1 discusses prior work related to the logic behavior derivation and physical localization of a back-end defect. The details of how PADLOC deduces the behavior and pinpoints the physical location of a defect are described in Section 4.2. Results from a simulation-based and silicon experiment are presented in Section 4.3, where it is demonstrated to be more effective in diagnosing a defect than the state-of-the-art. Finally, Section 4.4 concludes this chapter by highlighting the overall contributions of this work.

4.1 Prior Work

Several papers have been published over the years that improve the quality of diagnosis via back-end layout analysis[1]. For example, diagnosis techniques of [145, 147–155] are centered on open defects, while the techniques presented in [156, 157] focus on bridge defects. The approaches presented in [158–161] are applicable to both open and bridge defects, but do not focus on deriving the precise logic behavior of a defect.

The unpredictability of defects requires the use of a more generalized approach to defect diagnosis, or a multitude of "defect-based" approaches. But the disadvantage of using a set of defect-based approaches is the possibility that some defect behavior is missed. One general approach, referred to as DIAGNOSIX, is described in [142–144]. It is one of the first physically-aware diagnosis methods that is applicable to any defect type and handles defects

[1]Methods that improve the quality of diagnosis via front-end layout analysis are critiqued in Section 5.2.

exhibiting arbitrary misbehaviors as well. Instead of correlating observed defect behavior with fault models to characterize a defect, it derives the defect behavior by analyzing the logic activity of the nets surrounding its location.

Section 4.1.1 discusses defect-based approaches proposed in the literature and evaluates their merits and demerits. Section 4.1.2 describes DIAGNOSIX in detail, and lays the foundation of PADLOC.

4.1.1 Defect-based Approaches

Various techniques have been proposed in the literature that improve the diagnosis of open and bridge defects using circuit and layout parameters such as the state of the neighboring nets, coupling capacitances between a net and its neighbors, threshold voltages of the receiver standard cells and internal capacitances of the driver cells.

Techniques such as [158] extract features prone to bridge defects from layout using Design Rule Check (DRC) and Design for Manufacturability (DFM) guidelines. Potential bridge locations can also be derived using proximity analysis where a net can form a bridge with any net that is adjacent to it or within its line-of-sight [125, 142–144, 161, 278, 280]. Another approach is to find pairs of nets that are capacitively coupled via circuit parasitics. However, these approaches only focus on localizing a bridge defect, not identifying its behavior.

Work in [156] uses a voting bridge fault model [71] to diagnose a bridge defect. In the voting bridge fault model, the voltage at the shorted point is assumed to be influenced by the resistance of the pull-down and pull-up networks of the bridged nets. The voltage at the shorted point is then compared with the logic threshold of the standard cells driven by the bridged nets to determine if the voltage should be interpreted as logic-0 or logic-1. It should be noted that, in the voting bridge fault model, the logic threshold of each receiver gate is assumed fixed and identical (equal to $V_{DD}/2$). However, the logic threshold voltage of a cell can vary depending on the type of the cell and process node. The biased voting bridge fault

model [72] goes a step further and takes into account the fact that the voltage at the bridged point can be interpreted differently by each receiver cell.

However, both the voting bridge fault models are limited to modeling strong bridge defects, i.e., they don't model resistive bridges. Work in [157], on the other hand, is adept in diagnosing resistive bridges as well. That method employs the resistive bridge fault model [354] and considers the logic threshold voltage of each receiver cell, driving strength of the cells driving the bridged nets and bridged resistance to identify the defect behavior. However, the main drawback of such fault model based approaches is that the behavior of a defect is assumed to be restricted to the behaviors considered in the underlying fault model.

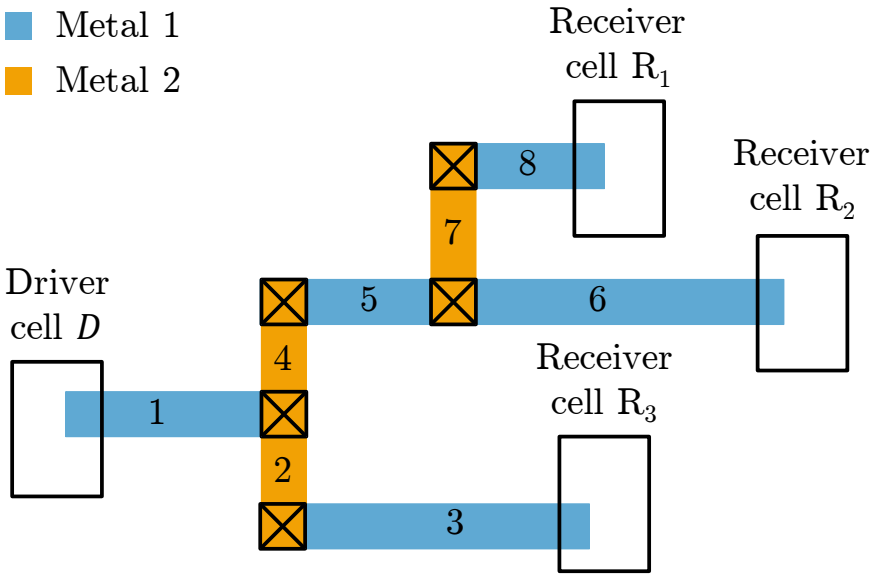
Additionally, extracting accurate parasitics may not be feasible, especially for advanced nanometer technologies. Specifically, as technology advances, an increase in design size and circuit density results in an increase in the number of parasitics and more importantly their complexity, which significantly impacts the performance of an electromagnetic field solver required to obtain accurate parasitics [355, 356]. Inaccurate estimation of the parasitics can thus lead to inaccurate characterization of the defect, which could subsequently misdirect PFA to inspect incorrect die locations, leading to a considerable amount of valuable resources being exhausted unnecessarily.

Regarding the diagnosis of an open defect, tracing a candidate net during PFA can be an expensive task depending on several factors including its length and number of metal layers it resides in. To improve localization of an open defect, each single via in the design is considered as a potential defect location in [158]. In [159], in addition to various via-related features such as single, multiple, stacked and stress vias, long nets having minimum width with nets on either side at minimum spacing [357] are also considered prone to open defects.

A "segment" model is presented in [147, 161], where each net is partitioned into segments such that each segment drives a different set of receiver cells. Partitioning in this way improves physical localization because only smaller parts of a net require examination via

PFA instead of the entire net. Moreover, net segmentation models an open defect more realistically. Figure 4.1 illustrates net segmentation to extract likely locations for an open defect. Figure 4.1 shows the topology of a net with three fan-out branches. The net is being driven by the standard cell, D, and drives three cells, R_1, R_2 and R_3.

Each net polygon in Figure 4.1 is numbered from 1 to 8. Each row in the table shown below the net topology represents a segment and consequently, an open defect location. The first column ("Segment") shows the polygons that comprise each segment. The second



Segment	Receiver cells driven
1	R_1, R_2, R_3
2, 3	R_3
4, 5	R_1, R_2
6	R_2
7, 8	R_1

Figure 4.1: Illustration of net partitioning into segments such that each segment drives a distinct set of receiver cells [147, 161].

column ("Receiver cells driven") shows the set of standard cells being driven by each segment. It can be seen that five open defect locations are identified using the technique of [147, 161]. Thus, net segmentation extracts likely locations of an open defect and improves physical localization by reducing the PFA search area from the entire net to one (or more) smaller net segments.

Work of [153] goes a step further in physically localizing an open defect, specifically an open via defect. Because a single net segment in [147, 161] could consist of multiple vias, work of [153] aims to pinpoint the exact via responsible for the observed failure by inspecting the nets that are physically close to its location. It should be noted that segments driving the same set of receiver cells but surrounded with a different set of physical neighbors are not differentiated.

Another segment model is proposed in [150] (and subsequently employed in [149,151,152] for diagnosis), where each net is divided into segments according to the topology of the physically adjacent nets. Figure 4.2 illustrates that segment model. It shows how a change in the topology of the surrounding nets partitions Net A into seven different segments.

However, in that work (and in [145]), various circuit-level parasitics are used to determine

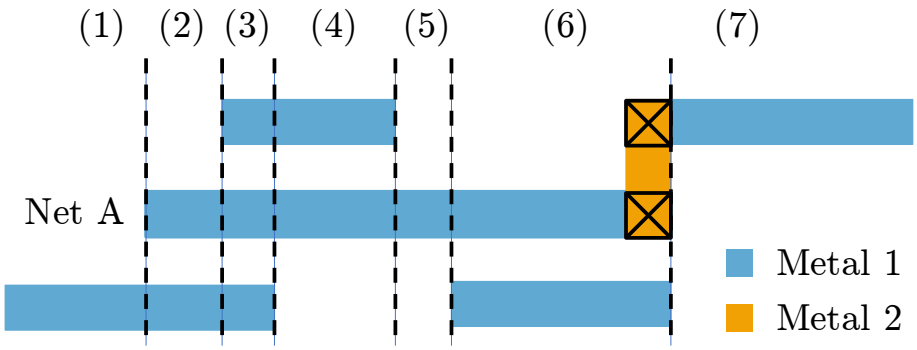


Figure 4.2: Net segmentation based on the topology of the net and its physical neighbors [149].

the voltage at the floating net. Specifically, it calculates the voltage from parasitic capacitance that exists between the floating net segment and its physically adjacent nets, and the trapped charge on the floating net. The technique of [148] improves upon [145,150] by taking into account the internal capacitances and the logic threshold voltages of the receiver cell(s). In addition to using these circuit parameters, the technique of [149] analyzes the effect of process variations while the work in [152, 358] incorporates gate leakage information to find the voltage of the floating net.

However, as reasoned earlier, methods that rely on approximate SPICE models to extract parasitics from a design are not desirable as the voltage estimated by them could be incorrect, especially at smaller process nodes, where the impact of variations induced by the manufacturing process and equipments is significant [359].

Another approach that focuses on the diagnosis of an open defect, which is described in [154,155], is based on the idea that the logic value at the candidate location is a (weighted) majority function of the logic values at its neighbors, where each weight is calculated based on the length of the neighbor and its distance from the candidate location. However, the effect of the logic threshold of the receiver cells is not taken into account.

In summary, prior work on defect-based approaches discussed up to this point focuses on localizing a defect and/or match the behavior of a defect candidate with a fault model (that could itself be inaccurate because of process variations and inexact SPICE models). But the limited fault models used only capture a subset of the behaviors exhibited by the known defects [360]. The arbitrary nature of a defect necessitates the need to instead derive the behavior of a defect with minimal assumptions. Work in [142–144] is a step in that direction. It is a more generalized approach and is discussed next.

4.1.2 DIAGNOSIX

DIAGNOSIX [142–144] is a comprehensive physically-aware diagnosis approach that finds potential defect locations without using a particular fault model, and derives a customized fault model that represents the defect behavior at each location. In that work, a candidate is assumed to be controlled by the nets surrounding its location. This observation is quite conservative and holds true for a variety of defects that includes bridge, open, and cell defects. The nets near the candidate are collectively referred to as its neighborhood, and includes all nets that are in physical proximity to the candidate, the inputs of the cell driving the candidate, and the side-inputs of the cells being driven by the candidate. The logic values established in the neighborhood of a candidate form its neighborhood state. The neighborhood state can also be dynamic in nature for sequence- and timing-dependent defects. Thus, in DIAGNOSIX, the excitation of a defect at candidate location is assumed to be a function of its neighborhood state.

Design layout is analyzed to find the physical neighbors of a net. Specifically, every neighboring net within some carefully-chosen distance d of the candidate is considered a physical neighbor of that candidate. Note that only intra-layer proximity is used to identify physical neighbors in that work.

In addition to physical neighbors, the inputs of the standard cell driving a candidate and the side-inputs of a standard cell being driven by the candidate are also considered to influence the logic value at the candidate location.

Figure 4.3 illustrates how DIAGNOSIX identifies the neighborhood of a net. Net A has two physical neighbors – net B and net C. Net C and net E drive net A, and D is the side-input of the cell being driven by net A. Thus, four nets are identified to be in the neighborhood of net A.

Neighborhood states for each candidate are collected for Tester-Fail-Simulation-Fail (TFSF) and Tester-Pass-Simulation-Fail (TPSF) patterns, and are referred to as failing and passing

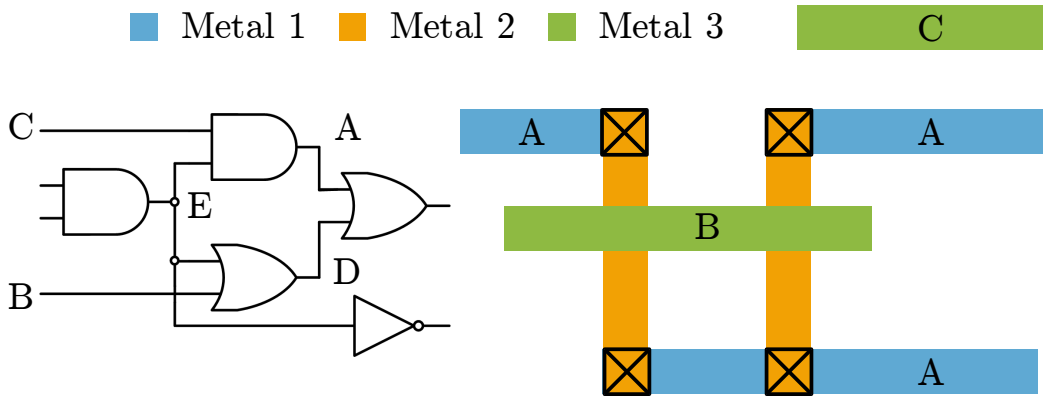


Figure 4.3: Example illustrating the neighborhood identified by DIAGNOSIX for net A

states, respectively. Here, a TFSF (TPSF) is a pattern that fails (passes) on the tester and detects either of the stuck-at faults at the candidate location. For a candidate to be deemed consistent, the sets of neighborhood states for TPSF and TFSF patterns should be disjoint. In other words, if the neighborhood state for a candidate is the same for any pair of TPSF and TFSF patterns, then the candidate has demonstrated inconsistency. In DIAGNOSIX, a candidate is eliminated from further consideration based on inconsistency. Consistency has been shown to be an effective method for improving resolution in [132, 142–144, 233, 306–308, 361].

Table 4.1 illustrates the consistency check for two candidates A and B. Assume that the neighborhood for each candidate consists of three nets. Column 1 shows the test pattern index and column 2 shows whether a pattern is TFSF or TPSF. Columns 3 and 4 show the neighborhood state for candidates A and B, respectively. It is evident from Table 4.1 that candidate B is consistent because the sets of failing and passing neighborhood states are disjoint. Candidate A, on the other hand, is inconsistent because a same neighborhood state is established for a TPSF and a TFSF pattern (specifically, T_1 and T_2, respectively).

Because it is assumed that the excitation of a defect candidate is controlled by its neighborhood, a customized fault model can be derived from the test data (example data is shown

Test pattern	Pass/fail pattern behavior	Neighborhood state for A	Neighborhood state for B
T_1	TPSF	000	100
T_2	TFSF	000	001
T_3	TFSF	001	010
T_4	TFSF	010	101
T_5	TFSF	010	110
T_6	TPSF	100	000
T_7	TPSF	100	011
T_8	TPSF	100	111

Table 4.1: Illustration of consistency check. Candidate A is inconsistent because the state 000 is established for a TPSF and a TFSF pattern. Candidate B is consistent because the failing states and the passing states are distinct.

in Table 4.1) that corresponds to the behavior of that candidate. In other words, the logical conditions that excite a defect can be identified. This is orthogonal to a fault model based approach where the candidate behavior is checked against the assumed model.

In DIAGNOSIX, the defect behavior is derived using Boolean minimization, where each state corresponding to a TFSF (TPSF) pattern is considered a "minterm" ("maxterm")[2]. For example, in Table 4.1, if the neighbors of B are denoted as N_1, N_2 and N_3, the behavior of the consistent candidate B is derived to be $N_2'N_3 + N_2N_3'$, which implies that B fails when N_2 and N_3 have opposite logic values. Accurate derivation of the defect behaviors affecting a population of failing chips can then be utilized to estimate the defect-type distribution of the population, which, in turn, likely accelerates PFA and yield learning.

DIAGNOSIX can be improved in several ways. First, the candidates reported by DIAGNOSIX are logical signals. However, the net that corresponds to the logical signal can be long and can span various metal layers. Examining a long net spread over a large area increases PFA effort. Thus, it is beneficial to improve the physical resolution [279] by identifying a

[2]A neighborhood state not exhibited by any pattern is treated as a "don't care".

net segment that is more likely to correspond to the actual defect location.

Second, not all neighbors of a net are relevant to defect excitation. Figure 4.4 illustrates this point for a bridge defect. Each segment in Figure 4.4 is assigned a name, and is denoted by 'segment name(net name)'. As shown, $S_5(N_2)$ and $S_6(N_3)$ are physical neighbors of $S_1(N_1)$ and $S_4(N_1)$, respectively. Consider a bridge defect (shown as a resistor) involving segments $S_1(N_1)$ and $S_5(N_2)$. DIAGNOSIX assumes that neighbor $S_6(N_3)$ can influence the faulty value of N_1, irrespective of the defect location. However, the logic value of $S_6(N_3)$ is less likely to influence the value at the bridged subnets because $S_6(N_3)$ is not physically close to the defect location. Net N_3 can affect the faulty value of N_1 only when there is a bridge defect involving $S_4(N_1)$. Thus, identification of relevant neighbors of a defect candidate can improve the physical resolution.

Similarly, Figure 4.5 illustrates this observation for an open defect. Consider an open defect located at $S_3(N_1)$, depicted as missing metal material with a red ✖. DIAGNOSIX assumes that the physical neighbors of N_1 will control the value of the open net. However, the value at the floating net can only be influenced by the physical neighbors downstream from the defect location. Thus, contrary to the assumption by DIAGNOSIX, $S_6(N_3)$ is

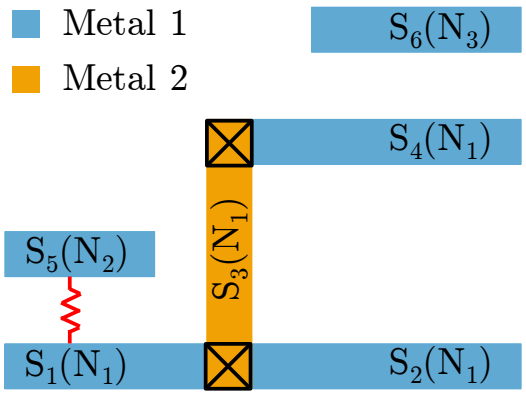


Figure 4.4: Example showing how all neighbors of a net, as identified from DIAGNOSIX, are not relevant to excite a bridge defect.

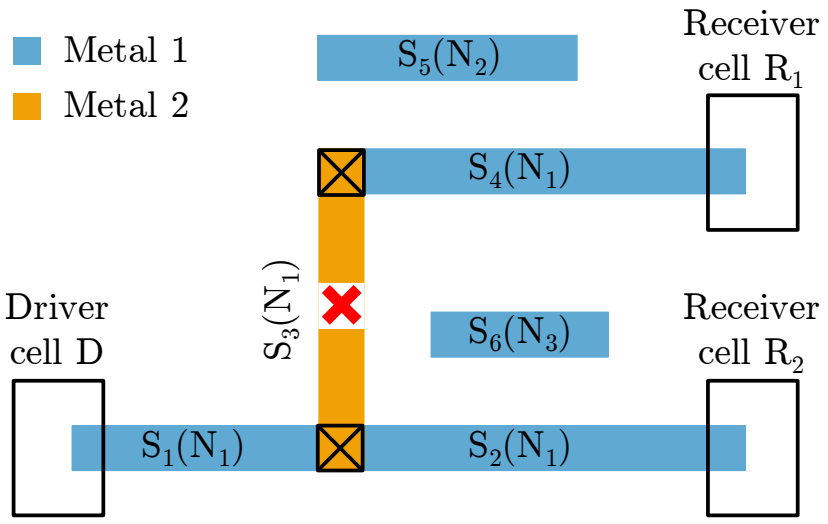


Figure 4.5: Example showing how all neighbors of a net, as identified from DIAGNOSIX, are not relevant to excite a bridge defect.

unlikely to play a role in deciding the value at the floating segment.

The first limitation of DIAGNOSIX can be addressed by partitioning a net into segments. Each segment would have its own set of neighbors. The partitioning would not only decrease the size of neighborhood, but also improve the physical resolution and further localize a defect. The criterion used for dividing each net into segments depends on the surrounding circuitry and is explained in the next section (Section 4.2). The choice of criterion would address the second shortcoming associated with DIAGNOSIX.

4.2 Diagnosis Methodology

The discussion of prior work associated with the diagnosis of back-end bridge and open defects using various layout and circuit parameters in Section 4.1 showed that no technique is proficient in the two main components of an ideal diagnosis approach simultaneously – physical localization and logic behavior derivation. PADLOC is a step in this direction.

Specifically, PADLOC has the following features.

1. It physically localizes a back-end defect using minimal layout information. A candidate net is partitioned into smaller subnets based on its topology and the topology of its physical neighbors to enhance physical localization.

2. Unlike prior work, it avoids the use of potentially inaccurate circuit parameters and parasitics to find the logic value at a defect location.

3. Instead of relying on traditional fault models or designing a new fault model, it derives the behavior of a defect based on its neighborhood. It monitors the logical activity on the surrounding nets to deduce the conditions that could produce the observed circuit response.

4. It is applicable to (localized) back-end defects that exhibit a variety of misbehaviors.

Figure 4.6 shows an overview of PADLOC. The input to PADLOC is a set of (back-end) defect candidates that have been identified by a physically-aware logic diagnosis methodology such as LearnX/MD-LearnX. The first step, referred to as *neighborhood-based net segmentation* in Figure 4.6, aims to physically localize a defect. Here, the topology of a defect candidate and its physical neighborhood are examined to pinpoint the layout area where the defect could reside. The next step, called *segment neighborhood identification*, identifies the subnets relevant to the activation of a defect. A *consistency check* is then performed on each candidate segment to identify segments that portray consistent pass-fail behavior. The amount of inconsistency may be adjusted to eliminate a candidate. The output of PADLOC is a set of candidate segments, where each candidate is described with respect to its defect type, logic behavior and physical location in the design.

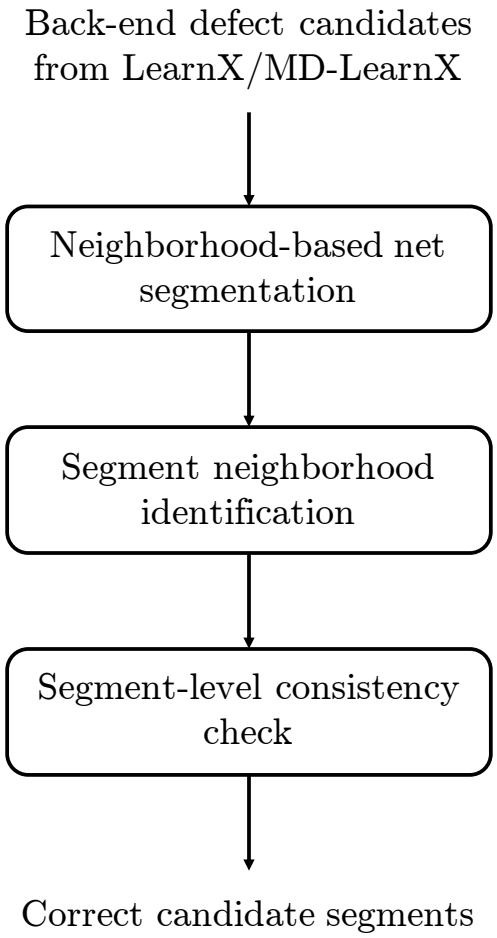


Figure 4.6: An overview of the proposed back-end diagnosis methodology, PADLOC.

4.2.1 Neighborhood-based Net Segmentation and Segment Neighborhood Identification

In PADLOC, to physically localize a bridge defect, each net is divided into segments such that each segment has its own set of physical neighbors. Two segments are said to be physically close to each other if (a) they are within some distance d[3] from each other, and (b) there is an unhindered line-of-sight between them.

[3] d is a user-defined parameter that depends on various factors including the metal interconnect pitch and the minimum metal layer width.

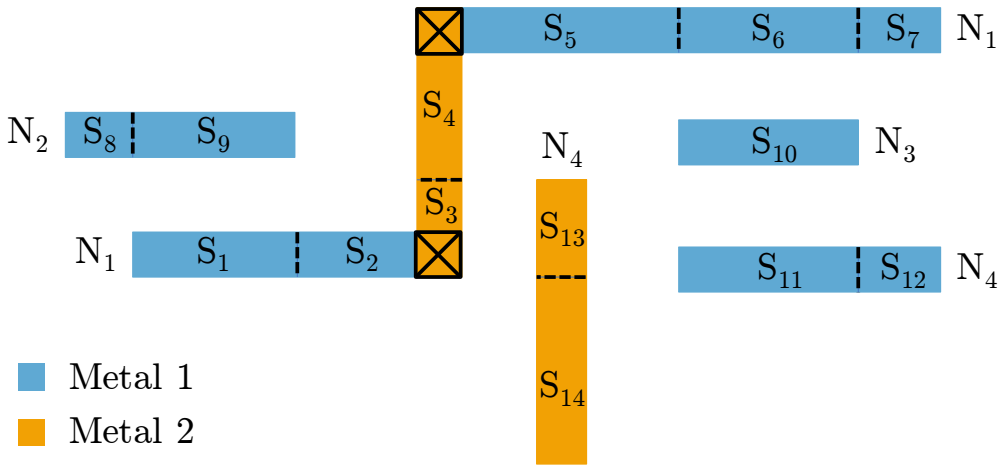


Figure 4.7: Illustration of net partitioning into segments for localizing bridge defects.

Segment	Physical neighborhood
S_1	S_9
S_2	-
S_3	S_3
S_4	-
S_5	-
S_6	S_{10}
S_7	-

Table 4.2: Physical neighborhood for each segment of net N_1 (illustrated in Figure 4.7) for bridge defect diagnosis.

Segment partitioning and neighborhood identification is illustrated in Figure 4.7. Each segment in Figure 4.7 is assigned a name S_i. The name of the net corresponding to each segment is shown besides the segment. Net N_1 is partitioned into seven segments $S_1 - S_7$ such that each segment has a different physical neighborhood. The physical neighborhood of each segment of N_1 is given in Table 4.2. It should be noted that although there is a line-of-sight between the segments S_7 and S_{12}, they may or may not be classified as physical neighbors, depending on the user-defined distance parameter d.

Given a bridge defect candidate, each net involved in the bridge defect can be partitioned

into different segments as illustrated in Figure 4.7. The layout region where a bridge defect likely resides is thus the area between each pair of physically close segments. For example, if logic diagnosis returns (N_1, N_2) as a possible bridge defect candidate, then it is likely that the defect is located in the layout region between segments S_1 and S_9. The implicated layout region can further be reduced via consistency check, which is discussed in Section 4.2.2.

To improve the physical localization of an open segment, a net is divided into segments such that (a) each segment drives a unique set of receiver standard cells, and (b) each segment has a unique physical neighborhood[4]. The physical neighborhood of each segment comprises of all the physical neighbors of the downstream segments and the segment itself. This is because the voltage at the floating node is not only influenced by the nets adjacent to it, but also (possibly) by the nets in the vicinity of the downstream segments.

Segment partitioning and neighborhood identification for net N_1 is illustrated in Figure 4.8. Net N_1 is being driven by cell D, and drives two cells, R_1 and R_2. Using the approach outlined above, PADLOC identifies eight segments $S_1 - S_8$ for net N_1. The physical neighborhood of each segment of N_1 is shown in Table 4.3. Note that the neighborhood of any segment upstream to an open defect location is unlikely to affect its value. For example, S_{11}, which is adjacent to S_5, is less likely to affect the logic value at segment S_3, S_4, S_6, S_7 or S_8.

Given an open defect candidate, the corresponding net can be divided into different segments as illustrated in Figure 4.8. By combining the net segmentation and neighborhood information from the pass-fail behavior of each receiver cell (that is known from a physically-aware logic diagnosis approach such as LearnX/MD-LearnX), relevant segments where the open defect could be present can be identified. The number of segments, and consequently the layout region, can further be reduced through consistency check, which is discussed in Section 4.2.2.

[4]It should be noted that a via that satisfies these conditions is also considered as a likely location of an open defect.

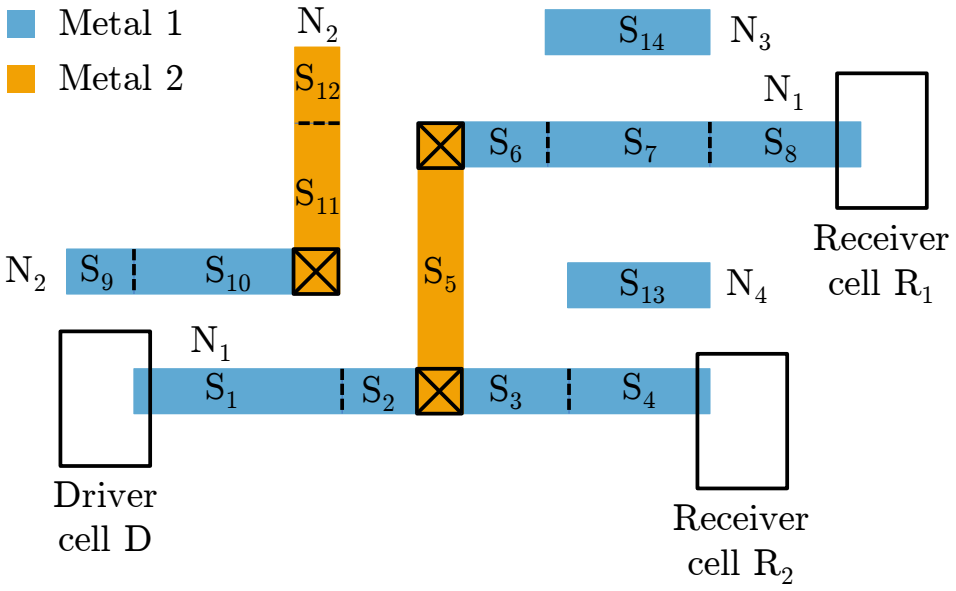


Figure 4.8: Illustration of net partitioning into segments for localizing open defects.

Segment	Physical neighborhood
S_1	S_{10}, S_{11}, S_{13}, S_{14}
S_2	S_{11}, S_{13}, S_{14}
S_3	S_{13}
S_4	S_{13}
S_5	S_{11}, S_{14}
S_6	S_{14}
S_7	S_{14}
S_8	-

Table 4.3: Physical neighborhood for each segment of net N_1 (illustrated in Figure 4.8) for open defect diagnosis.

Thus, by breaking up a net into segments based on the topology of its physical neighborhood and the net itself, PADLOC identifies relevant segments that most likely relate to a defect and influence its excitation.

Recall that there are two components of a neighborhood of a net – physical and logical. In DIAGNOSIX, the logical neighborhood of a net consists of the inputs of the cell driving the net and the side-inputs of the cells being driven by the net. However, the logical neighbors

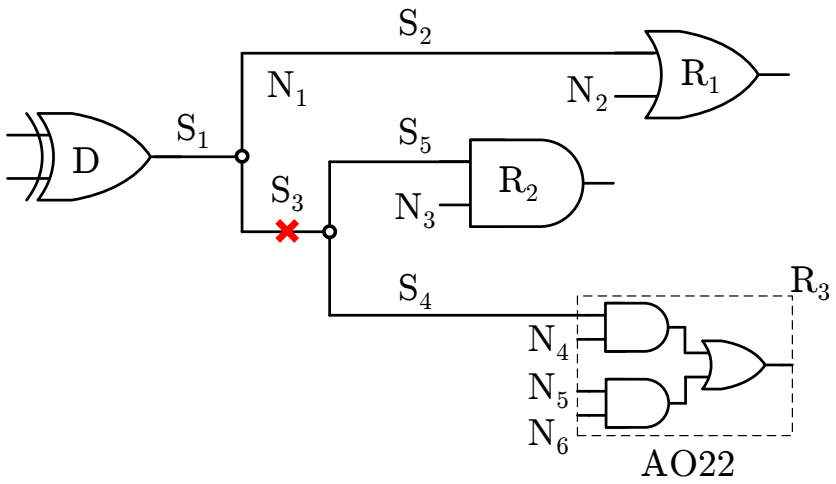


Figure 4.9: An example defective circuit illustrating the identification of relevant logical neighbors.

play different roles for different defect types. For example, the voltage of bridged nets depends on the logic values of the inputs of the cells driving the bridged nets and the state of the surrounding circuit, and is not influenced by the side-inputs of the receiver cells [72].

An example circuit shown in Figure 4.9 illustrates the process of identifying logical neighbors that are more likely to be involved with an open defect. Suppose that the topology of net N_1 in the layout of the circuit is similar to that shown in the schematic in Figure 4.9. Further suppose that partitioning the net associated with an open defect candidate results in five segments, $S_1 - S_5$. For an open defect (strong open) at segment S_3 (net N_1), two observations are made. First, the inputs of cell D cannot influence the logic value at the floating part of S_3 because the output of D is disconnected from it. Second, as S_3 drives the cells, R_2 and R_3, only side-inputs of those two cells would form its logical neighborhood. Thus, in this case, the logical neighborhood of S_3 consists of the nets $\{N_3, N_4, N_5, N_6\}$.

Furthermore, each side-input of a basic cell such as AND, NAND, OR, NOR should have a non-controlling logic value to propagate an error due to an open defect to the cell output. This reasoning can be extended to non-basic cells as well. For example, for the AO22 cell

(that has been mapped to basic gates) shown in Figure 4.9, N_4 can be discarded as a logical neighbor of S_3 because it is one of the inputs of the AND cell (within R_3) and its value should be logic-1 to propagate the error through that cell. However, the other two side-inputs of cell R_3, N_5 and N_6, remain as logical neighbors of S_3 because either one of them (or both) can be logic-0 to propagate the error through the OR cell to the output of R_3.

Thus, for the defective circuit shown in Figure 4.9, the logical neighborhood of segment S_3 consists of N_3, N_5 and N_6. On the other hand, DIAGNOSIX identifies nets N_2, N_3, N_4, N_5 and N_6 along with the inputs of cell D as the logical neighbors. Thus, in addition to the decrease in the size of the neighborhood (which means fewer nets need to be monitored), nets pertinent to activating an open defect are deduced. Identifying the neighborhood that most likely influences the logic value at the defect location is important from a yield perspective as well. Defect behaviors from diagnosing numerous failing chips can be correlated to estimate the probability of a defect exhibiting a specific misbehavior, which can then be used to employ adaptive testing, and correct design, test and manufacturing issues to improve yield.

4.2.2 Segment-level Consistency Check

After determining the neighborhood for each candidate segment, the next step in PADLOC is deriving its neighborhood states for the TFSF and TPSF patterns. The notion of obtaining the neighborhood states and performing the consistency check that is pioneered in DIAGNOSIX for a logical signal (and described in Section 4.1.2) is adapted here for a (physical) net segment.

In PADLOC, a neighborhood state for a segment consists of the set of logical values established by a pattern on the segments in its neighborhood. Neighborhood states obtained for the TFSF patterns are called the failing states, while the states identified for the TPSF patterns are called the passing states. Because the underlying assumption in PADLOC is that the logic value at a defect location is a function of its neighborhood, a segment is deemed

inconsistent if the same neighborhood state occurs for a pair of TFSF and TPSF patterns. Mathematically, a consistent segment has disjointed sets of failing and passing states. Based on the amount of inconsistency, i.e., the number of neighborhood states that are failing as well as passing, a segment could be eliminated as a candidate. Each candidate can also be assigned a score based on the size of its neighborhood, the number of failing and passing states, etc. Candidate ranking is further explored in Chapter 6.

4.3 Experiments

This section describes two experiments to validate PADLOC: one is a defect injection and simulation experiment using four different designs (Section 4.3.2) and another that uses real silicon failure data (Section 4.3.3). The diagnostic metrics that are used to quantify the effectiveness of PADLOC are discussed in Section 4.3.1.

In each experiment, each fail log is analyzed using PADLOC as follows. The input to PADLOC is a set of defect candidates reported by LearnX using the steps outlined in Figure 2.2. For each fail log, segments and their corresponding neighborhoods are identified for each defect candidate. Passing (failing) neighborhood states are derived for each candidate and for each TPSF (TFSF) pattern. Finally, a candidate segment is deemed correct if it is found to be consistent. For a bridge defect, the output of PADLOC is the set of layout regions between two consistent, physically-close segments. For an open defect, PADLOC reports the layout region corresponding to each consistent segment.

4.3.1 Diagnostic Metrics

As noted in Section 1.2, a diagnosis methodology should accomplish two objectives. First, it should accurately pinpoint the area where the suspected defect resides. Second, it should identify the type of a defect and derive its precise logic behavior.

PADLOC focuses on accomplishing each objective[5]. Note that, while PADLOC characterizes a defect in terms of its logic behavior, the effectiveness of logic characterization of a defect can only be realized in subsequent failure analysis, and remains the focus of the future work. For instance, a statistically-significant volume of diagnoses can be effectively correlated to determine the defect behavior distribution, if the logic behavior of a defect candidate is precisely derived. As a result, the underlying root cause of yield loss can be identified without any significant PFA effort, which, consequently, propels yield learning.

Here, the performance of PADLOC is compared with two advanced commercial diagnosis tools and evaluated using the following three metrics.

1. Physical accuracy: This metric measures whether a defect resides in the layout region suspected by a diagnosis approach.

2. *Bounding circle diameter* [161]: *Bounding circle diameter* quantifies the physical diagnostic resolution for a circuit (and the corresponding test set). Here, *bounding circle diameter* is defined as the diameter of the smallest circle that contains the layout region suspected by diagnosis. It estimates the area over which the candidates are distributed in the layout. In [161], it is argued that the cost of performing PFA is directly proportional to the layout area over which the candidates are scattered. Thus, this metric can directly be used to determine if a failing chip is suitable for PFA.

3. *Area union* [279]: This metric quantifies physical resolution as well. It calculates the area of the geometric union of the layout region suspected for each candidate segment. It subsumes any overlapping area among candidates.

Both the physical resolution metrics are normalized with respect to the metal-1 pitch of a technology process.

[5]The goal of correctly identifying the type of a defect is attained by LearnX and MD-LearnX.

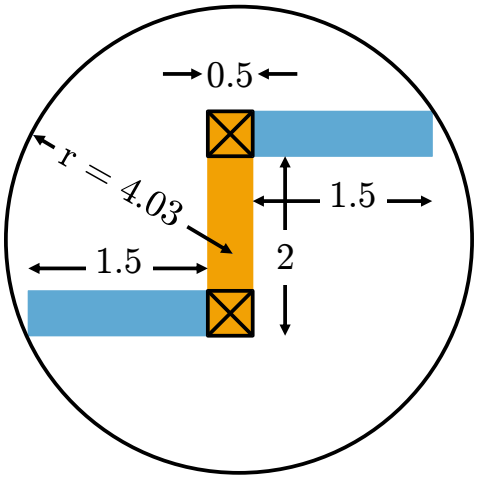


Figure 4.10: A simple example illustrating the computation of two diagnostic metrics, *bounding circle diameter* and *area union*.

Figure 4.10 illustrates the computation of *bounding circle diameter* and *area union*. Suppose that the layout region implicated by PADLOC is a set of three segments (shown in color). The dimensions of each segment and the enclosing circle are shown in the figure. The two metrics, *bounding circle diameter* and *area union*, are calculated as follows.

$$\textit{Bounding circle diameter} = (4.03)(2) = 8.06$$

$$\textit{Area union} = (1.5)(0.5) + (2)(0.5) + (1.5)(0.5) = 2.5$$

4.3.2 Simulation Experiment

The setup for the comprehensive defect injection and simulation experiment employed to evaluate PADLOC is described in Section 4.3.2.1. In Section 4.3.2.2, the quality of diagnosis achieved by PADLOC is compared with LearnX to study the improvement in physical resolu-

tion and effect on physical accuracy. To gauge the performance of PADLOC, it is compared with state-of-the-art commercial diagnosis (Section 4.3.2.3). Section 4.3.2.4 summarizes the experiment results to emphasize the improvement in diagnosis quality attained by PADLOC.

4.3.2.1 Setup

For validating PADLOC, a simulation-based experiment is performed using four different designs. One is an Advanced Encryption Standard (AES) core that provides AES-128 encryption (henceforth referred to as "AES") [311], the second design is an IWLS'05 benchmark that implements the Data Encryption Standard (DES) algorithm called *des_perf* (henceforth referred to as "DES"), the third design is the L2 cache of the OpenSPARC T2 processor (called "L2B") [312], and the fourth design is an ITC'99 benchmark circuit called "B18" [313]. Each circuit is synthesized, and placed and routed using a 45nm technology library [314].

To mirror a real scenario, a pool of defective circuits is created by injecting bridge and open defects into the layout of each design. The location of a bridge defect is identified using geometric proximity analysis[6]. An injected bridge can be a bridge between two nets (or vias) that run parallel to each other or a bridge between the corners of two adjacent nets (or vias). In addition to injecting net open defects, different types of vias such as single vias, multiple vias and stacked vias are considered as possible locations for injecting an open defect.

Virtual fail logs are generated using a mixed-signal simulation tool called SLIDER [362, 363]. In SLIDER, a circuit-level simulation is performed for the circuitry surrounding a defect, while a logic-level simulation is performed for the rest of the circuit. For each design and for each defect type (i.e., bridge and open), 1,000 virtual fail logs are produced. The test set that is used to simulate each defective circuit is generated using a commercial ATPG software.

[6]Likely bridge locations can also be derived using Design Rule Check (DRC) rules, Design-for-Manufacturability (DFM) guidelines and capacitive coupling methods.

4.3.2.2 Comparison with LearnX

In this experiment, the quality of diagnosis achieved by PADLOC is compared with LearnX [274]. LearnX is a physically-aware diagnosis methodology that reports defect candidates at a logic level. For a candidate of type BRIDGE, LearnX reports the corresponding pair of bridged nets along with the layout region where the nets are physically close to each other. PADLOC, on the other hand, reports the layout region between every pair of consistent segments that are adjacent. For a candidate of type OPEN, LearnX divides the corresponding net into partitions such that each partition drives a unique set of receiver cells, and identifies the partition that could explain the observed circuit response. PADLOC further divides each partition into segments based on the topology of the neighborhood, performs a consistency check on each segment and reports a set of consistent segments at the end. Because PADLOC focuses on the layout analysis of back-end defect candidates, candidates of type CELL reported by LearnX are examined using another approach called NOIDA [162], which is the focus of Chapter 5.

Figures 4.11, 4.12, 4.13 and 4.14 plot the probability density distribution of *bounding circle diameter* and *area union* for AES, DES, L2B and B18, respectively, when bridge defects are diagnosed by PADLOC and LearnX.

A density plot can be thought of as a smoothed histogram. Thus, a wider distribution at a particular value indicates more density at that value. In each figure, the y-axis shows the distribution of *bounding circle diameter* and *area union* for each diagnosis technique shown on the x-axis. A dashed line across each density plot shows the mean of the distribution, while a dotted line shows the median. The physical accuracy, defined in Section 4.3.1, attained by each diagnosis approach is displayed at the top of each figure.

Observations specific to each design regarding bridge defect diagnosis are as follows.

1. **AES**: (Figure 4.11) The average physical accuracy for PADLOC is 98.1%, which is

the same as LearnX. The mean (median) diameter of the bounding circle for PADLOC is 21.2 (4.4), which is 4.7% (1.3%) better than LearnX. In addition, PADLOC reports smaller *bounding circle diameter* for 2.0% of fail logs. The mean (median) *area union* for PADLOC is 12.4 (3.7), which is 11.1% (4.7%) better than LearnX. Moreover, PADLOC reports smaller *area union* for 2.7% of fail logs.

2. **DES**: (Figure 4.12) PADLOC and LearnX attain an average physical accuracy of 97.1%. The mean *bounding circle diameter* for PADLOC is 22.0, an improvement of 13.1% over LearnX. Moreover, PADLOC reports smaller *bounding circle diameter* for 3.8% of fail logs. The mean *area union* for PADLOC is 22.2, which is 13.1% better than LearnX. Moreover, PADLOC reports smaller *area union* for 4.6% of fail logs.

3. **L2B**: (Figure 4.13) The physical accuracy achieved by PADLOC and LearnX is 96.7%. The mean *bounding circle diameter* for PADLOC is 32.9, an improvement of 4.2% over LearnX. In addition, PADLOC reports a smaller enclosing circle for 2.1% of fail logs. The mean (median) *area union* for PADLOC is 35.2 (7.6), which is 15.1% (4.0%) better than LearnX. Moreover, PADLOC reports smaller *area union* for 2.4% of fail logs.

4. **B18**: (Figure 4.14) Both LearnX and PADLOC locate the defect correctly for 97.5% of fail logs. The mean *bounding circle diameter* for PADLOC is 16.1, which is 11.2% less (i.e., better) than LearnX. Additionally, PADLOC reports a smaller bounding circle for 2.4% of fail logs. The mean (median) *area union* for PADLOC is 11.2 (3.4), where the improvement over LearnX is 17.9% (3.4%). In addition, PADLOC reports smaller *area union* for 2.6% of fail logs.

The following common observations can be drawn regarding the diagnosis of a bridge defect (Figures 4.11-4.14).

1. For each design, it is observed that the average physical accuracy achieved by PADLOC

and LearnX is identical when a bridge defect is diagnosed. This is because a pair of bridged segments returned by LearnX always passes the consistency check. Thus, if LearnX is able to identify the correct bridge defect, PADLOC cannot eliminate the segments associated with the defect.

2. Because diagnosis is not perfect, an open defect candidate (in addition to the correct bridge candidate) can sometimes be returned by LearnX while diagnosing a bridge defect. Such an open candidate (or, one or more of its segments) can be deemed inconsistent by PADLOC, which could result in an improvement in the physical resolution. Thus, PADLOC not only improves the physical resolution of a failing chip affected by a bridge defect without losing accuracy, but also (possibly) eliminates a candidate of an incorrect defect type, and thus instilling more confidence in the diagnosis result.

Figures 4.15-4.18 compare the performance of PADLOC and LearnX for open defects. In general, compared to LearnX, a slight loss in accuracy results in a significant improvement in the average physical resolution for PADLOC, which is reflected in the shapes of the distribution of *bounding circle diameter* and *area union*. Figures 4.15 through 4.18 reveal the following for each design.

1. **AES**: (Figure 4.15) The average physical accuracy for PADLOC is 97.6%, which is 1.3% less than LearnX. The mean (median) *bounding circle diameter* for PADLOC is 99.3 (106.5), which is 24.3% (27.6%) better than LearnX. In addition, PADLOC reports smaller *bounding circle diameter* for 42.6% of fail logs, where the highest reduction in *bounding circle diameter* over LearnX is 99.4%. The mean (median) *area union* for PADLOC is 70.9 (25.2), which is 39.6% (80.0%) improvement over LearnX. Moreover, PADLOC implicates smaller *area union* than LearnX for 46.6% of fail logs, with LearnX reporting *area union* at most 500X times PADLOC.

2. **DES**: (Figure 4.16) PADLOC attains an average accuracy of 97.5%, which is 1.4% less than LearnX. The mean (median) *bounding circle diameter* for PADLOC is 77.0 (37.2), which is 17.0% (22.7%) less (i.e., better) than LearnX. Additionally, PADLOC reports a smaller bounding circle for 33.0% of fail logs, where the maximum improvement in *bounding circle diameter* over LearnX is 98.9%. The mean (median) *area union* for PADLOC is 36.1 (15.6), which is 28.4% (43.0%) better than LearnX. In addition, PADLOC reports smaller *area union* than LearnX for 39.3% of fail logs, where the maximum decrease in *area union* is 99.2%.

3. **L2B**: (Figure 4.17) The physical accuracy achieved by PADLOC is 92.4%. PADLOC diagnoses 1.9% fewer fail logs correctly when compared with LearnX. The mean (median) *bounding circle diameter* for PADLOC is 131.8 (49.6), an improvement of 6.8% (24.1%) over LearnX. In addition, PADLOC reports smaller *bounding circle diameter* for 34.6% of fail logs, and reduces *bounding circle diameter* by up to a factor of 65. The mean (median) *area union* for PADLOC is 143.9 (21.8), which is 17.5% (44.0%) better than LearnX. Moreover, PADLOC reports smaller *area union* than LearnX for 38.6% of fail logs, where *area union* is decreased by at most 100X.

4. **B18**: (Figure 4.18) PADLOC correctly locates the defect for 95.4% of fail logs, which is 1.6% less than LearnX. The mean (median) *bounding circle diameter* for PADLOC is 95.6 (32.2), which is 23.0% (58.5%) better than LearnX. Additionally, PADLOC reports a smaller bounding circle than LearnX for 44.9% of fail logs, with the maximum decrease in *bounding circle diameter* being 99.7%. The mean (median) *area union* for PADLOC is 60.1 (11.7), which is 39.7% (71.4%) better than LearnX. In addition, PADLOC reports smaller *area union* than LearnX for 48.4% of fail logs, and decreases *area union* by up to a factor of 625.

The performance of PADLOC primarily depends on the effectiveness of two steps, namely,

neighborhood identification and consistency check. The possibility of how a correct candidate can be eliminated by either of these steps is as follows.

1. **Neighborhood identification**: The physical neighborhood of a segment is identified using geometric proximity and line-of-sight analysis. It is possible that the derived neighborhood does not capture all the defect activation conditions. One of the ways this can happen is that the distance d for searching a physical neighbor of a candidate, which is possibly determined by technology and design parameters, is not carefully selected. For example, its value is chosen to be equal to two times the minimum metal width in [142–144] and five times in [150].

   If d is too small, a portion of the neighborhood relevant to defect excitation might not be included, which in turn could result in the elimination of a correct candidate. On the other hand, if the size of the neighborhood is too large, it may result in the identification of neighbors that are irrelevant, which in turn could result in incorrect derivation of defect behavior and consequently, incorrect deduction of the failure root cause.

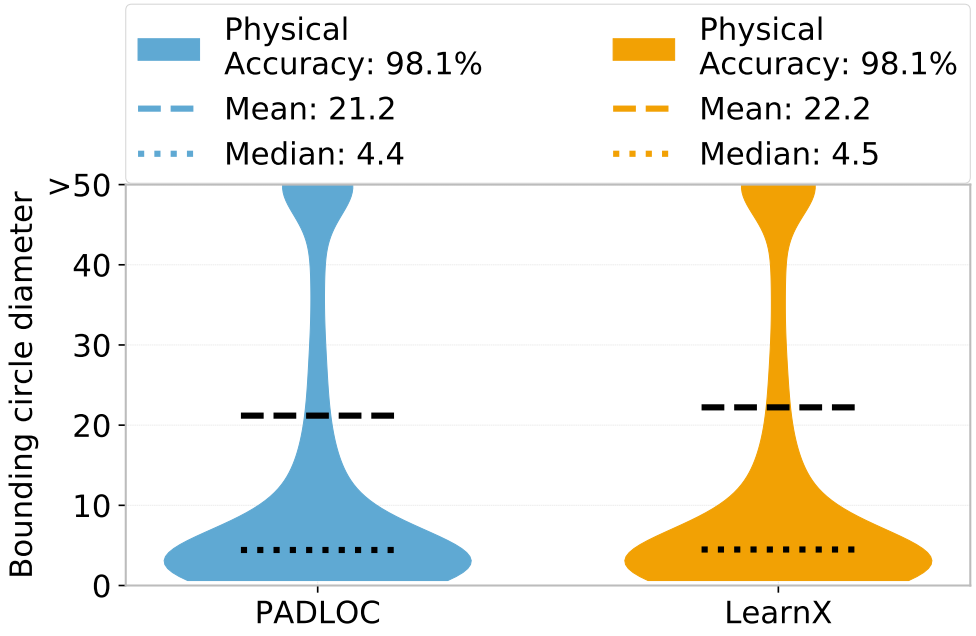
2. **Amount of (in)consistency**: It should be recalled that a candidate is eliminated from further analysis based on the cardinality of the intersection set of passing and failing neighborhood states. An aggressive heuristic would discard a candidate if the cardinality of the intersection set is non-empty; a conservative heuristic would remove a candidate if the set of failing states is a subset of the passing states.

   Here, an aggressive heuristic is employed, which results in a significant improvement in the physical resolution, albeit with a slight loss in physical accuracy. An alternative is to instead derive a data-driven heuristic that exploits machine learning to find the right balance between physical accuracy and resolution.

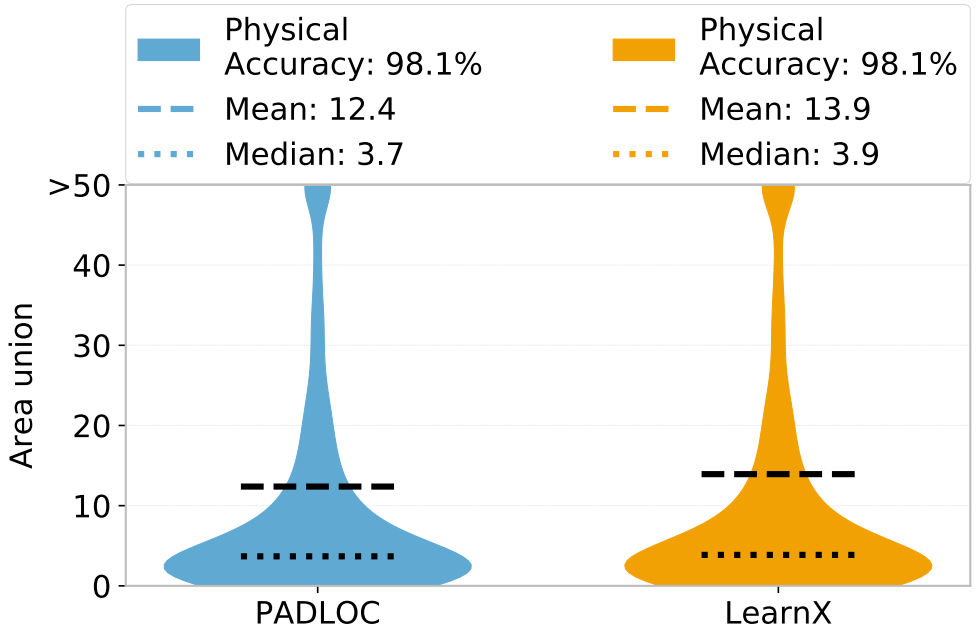
Improving the performance of PADLOC by focusing on these two aspects remains as future work and is further discussed in Chapter 6.

In summary, PADLOC correctly finds the defect location for 96.5% of the fail logs, when averaged over different designs. The enhancement in the physical resolution for PADLOC is substantial, albeit with a slight loss in its physical accuracy of 0.8%. The mean (median) diameter of the bounding circle for PADLOC is 15.6% (28.7%) better than LearnX. PADLOC reports a smaller bounding circle for 20.1% of fail logs, while decreasing *bounding circle diameter* by up to a factor of 300. When compared to LearnX, PADLOC improves the mean (median) *area union* by 26.4% (37.0%). PADLOC implicates smaller *area union* for 22.5% of fail logs, where *area union* is decreased by at most 625X.

On average, the runtime performance overhead for PADLOC is 0.35s per fail log, which means PADLOC adds less than 5% runtime overhead to LearnX. The runtime overhead can be decreased by storing the logic state of the internal nets beforehand, which can, however, be achieved at the cost of memory.

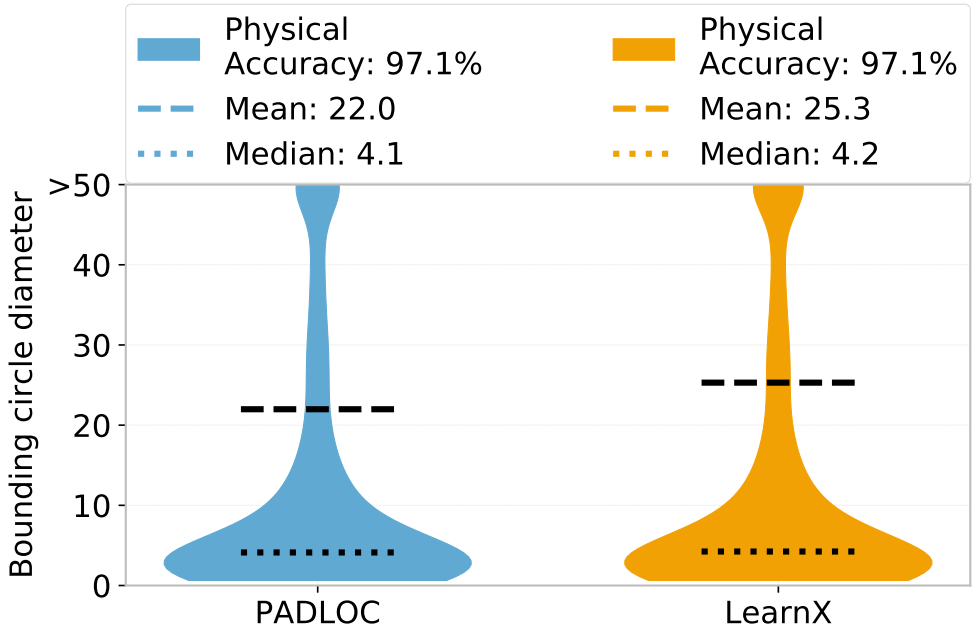


(a)

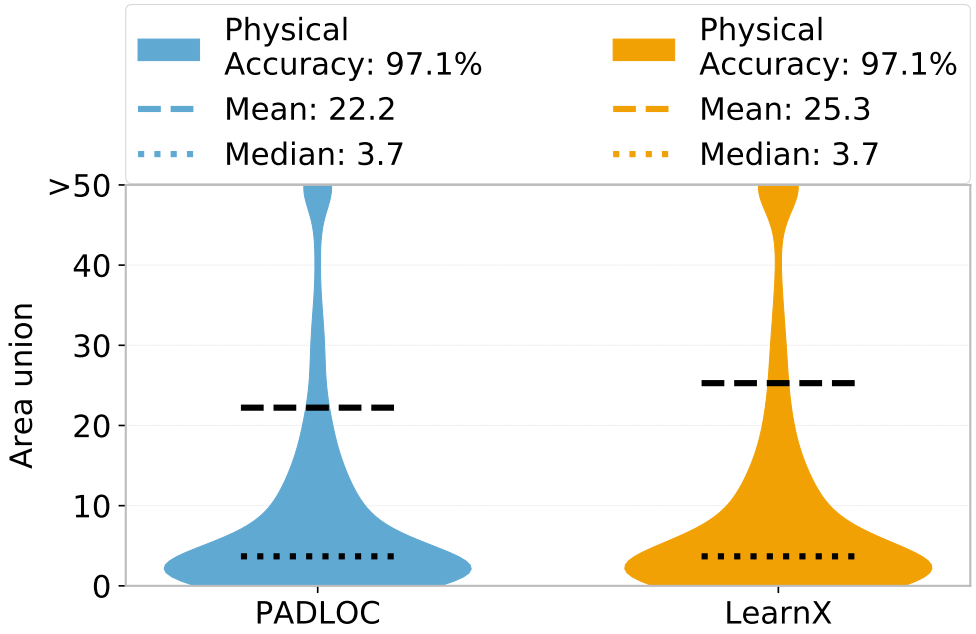


(b)

Figure 4.11: Probability density distribution of (a) *bounding circle diameter* and (b) *area union* for the design "AES", when bridge defects are diagnosed by PADLOC and LearnX.

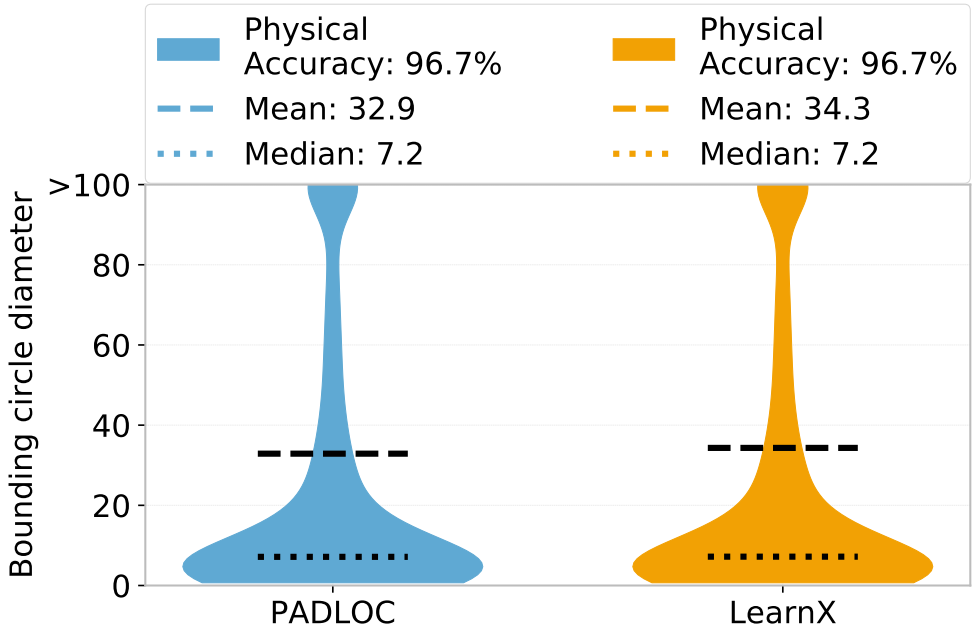


(a)

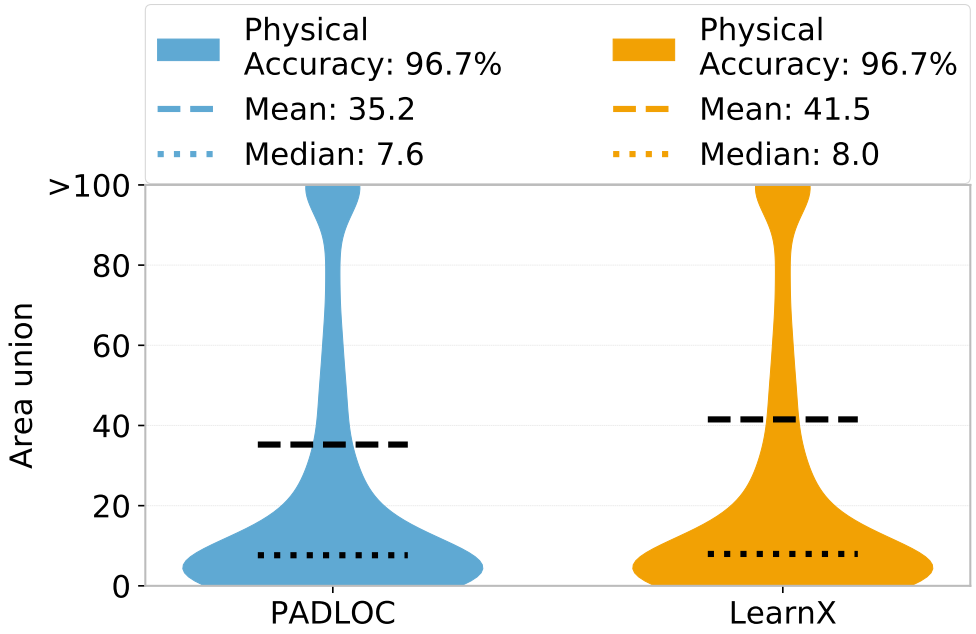


(b)

Figure 4.12: Probability density distribution of (a) *bounding circle diameter* and (b) *area union* for the design "DES", when bridge defects are diagnosed by PADLOC and LearnX.

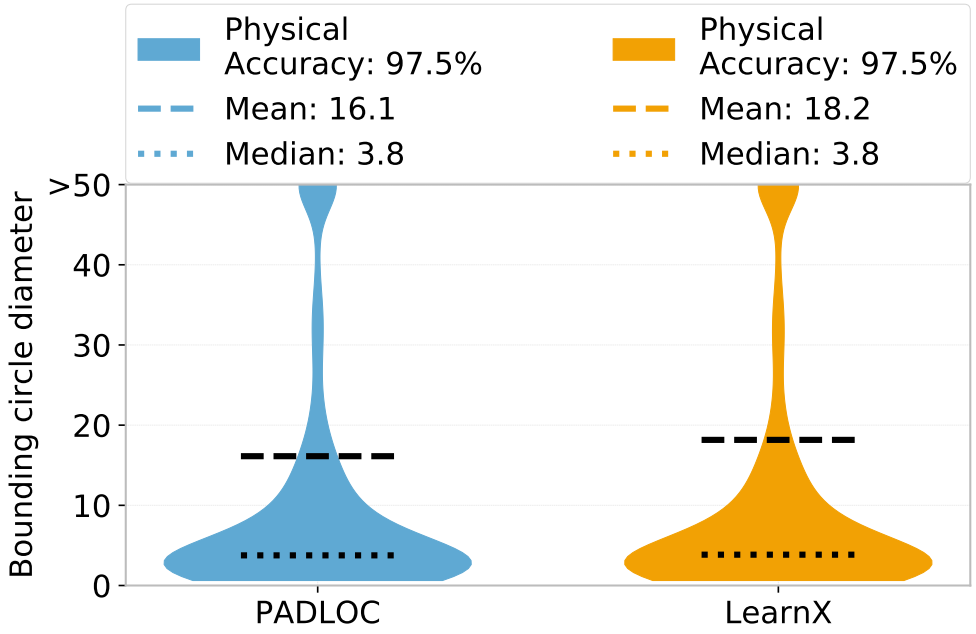


(a)

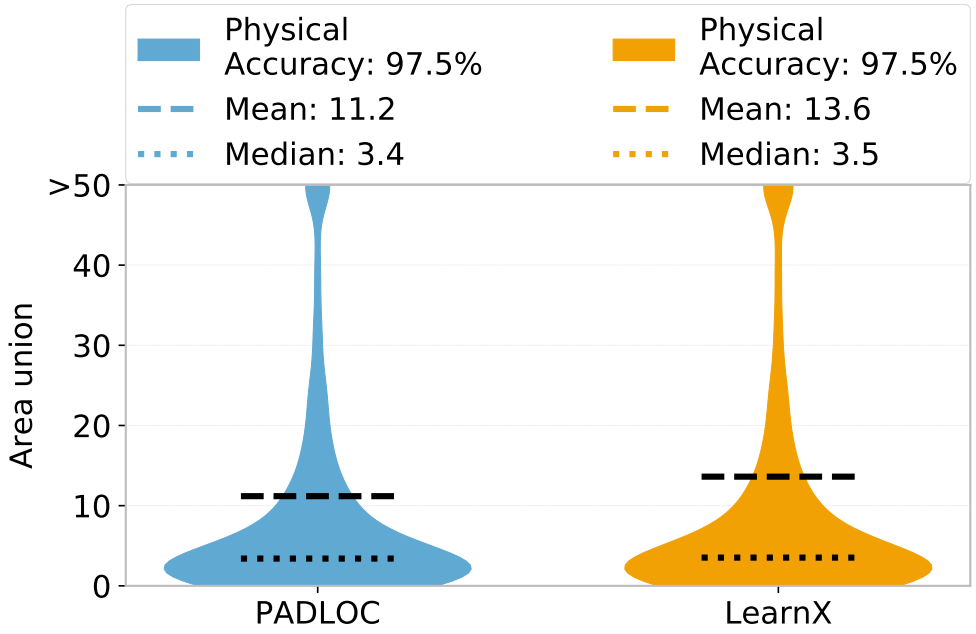


(b)

Figure 4.13: Probability density distribution of (a) *bounding circle diameter* and (b) *area union* for the design "L2B", when bridge defects are diagnosed by PADLOC and LearnX.

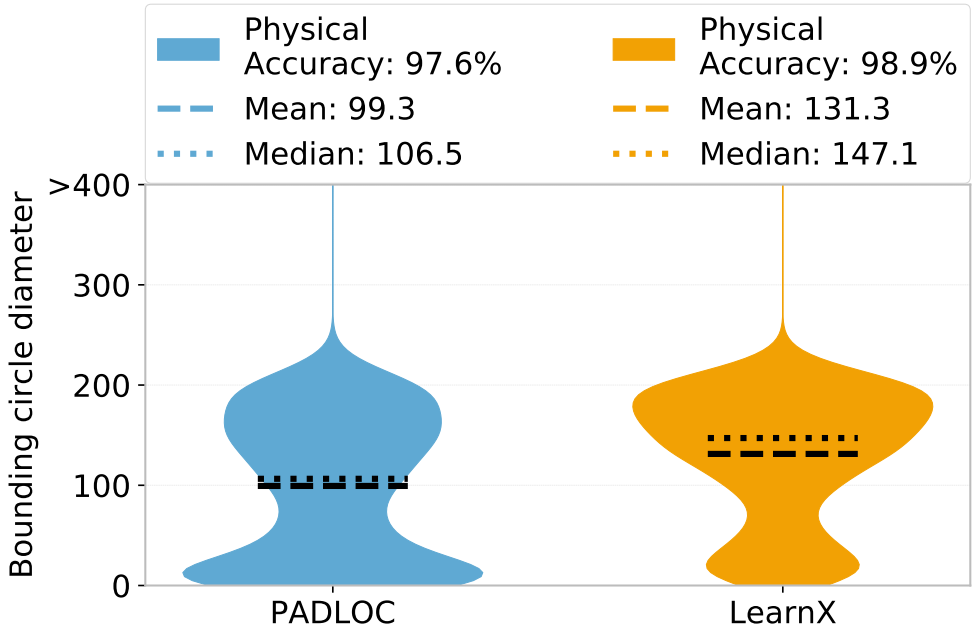


(a)

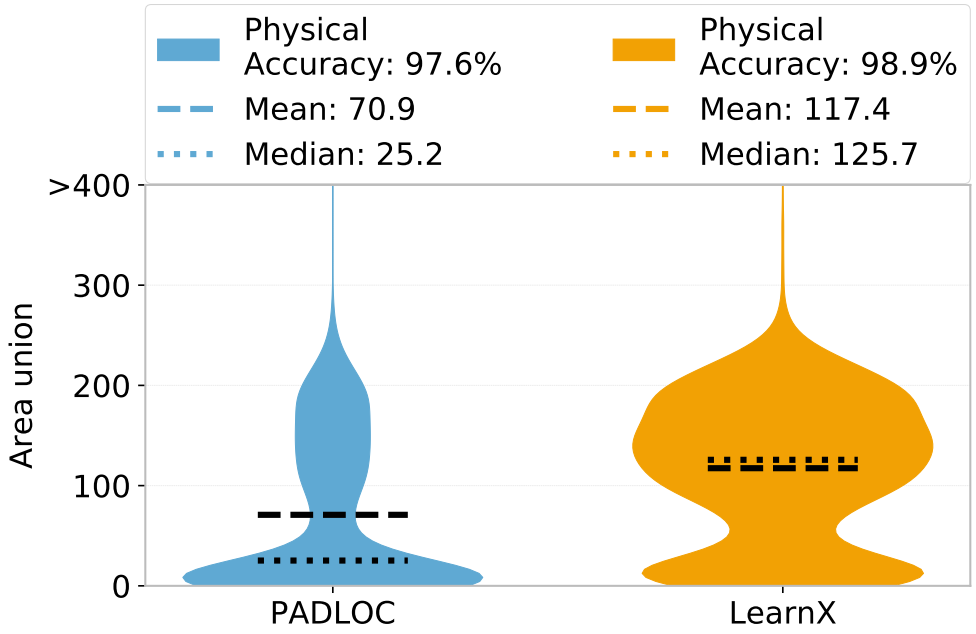


(b)

Figure 4.14: Probability density distribution of (a) *bounding circle diameter* and (b) *area union* for the design "B18", when bridge defects are diagnosed by PADLOC and LearnX.

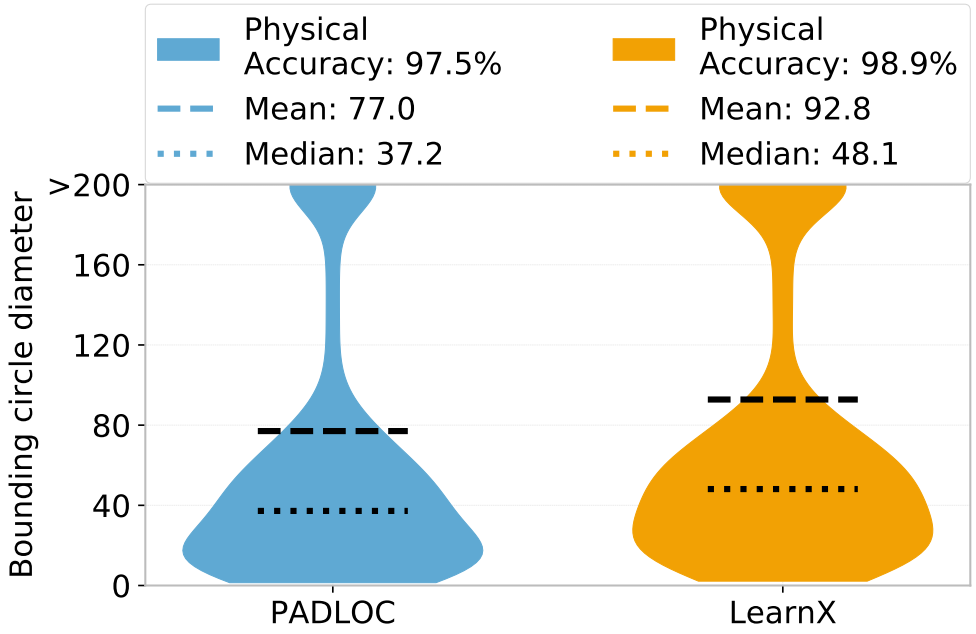


(a)

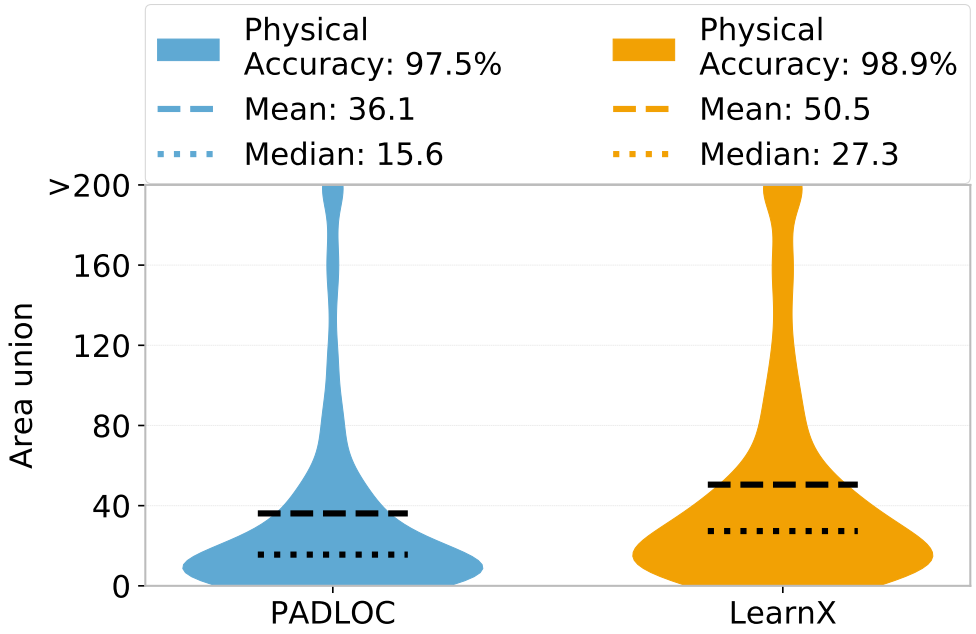


(b)

Figure 4.15: Probability density distribution of (a) *bounding circle diameter* and (b) *area union* for the design "AES", when open defects are diagnosed by PADLOC and LearnX.

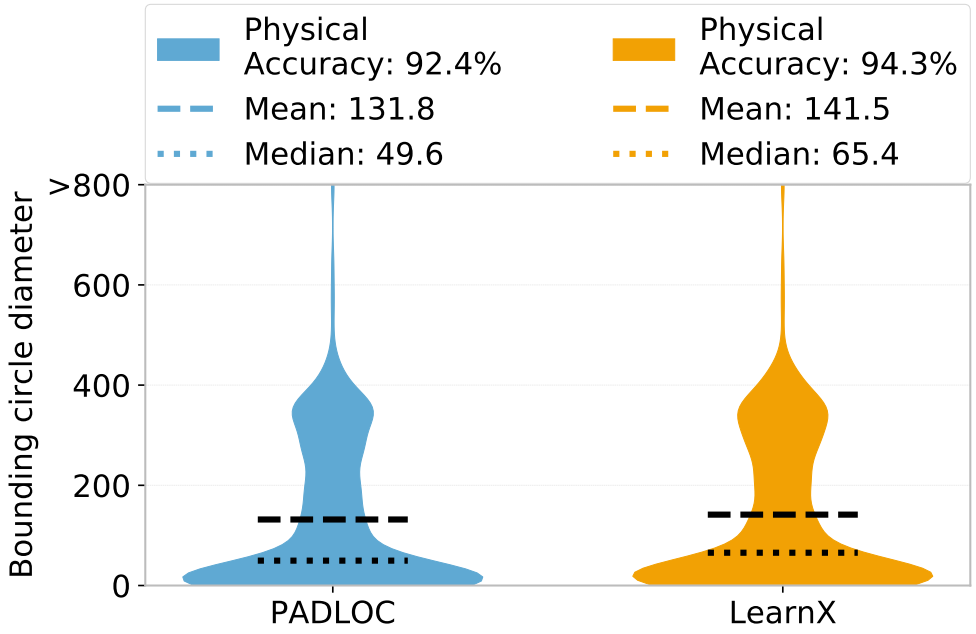


(a)

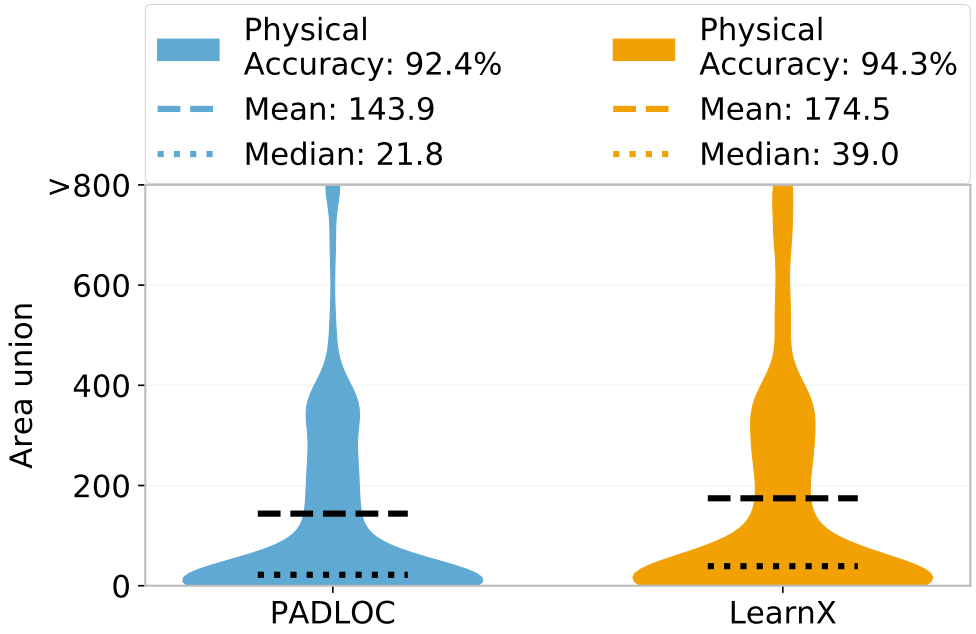


(b)

Figure 4.16: Probability density distribution of (a) *bounding circle diameter* and (b) *area union* for the design "DES", when open defects are diagnosed by PADLOC and LearnX.

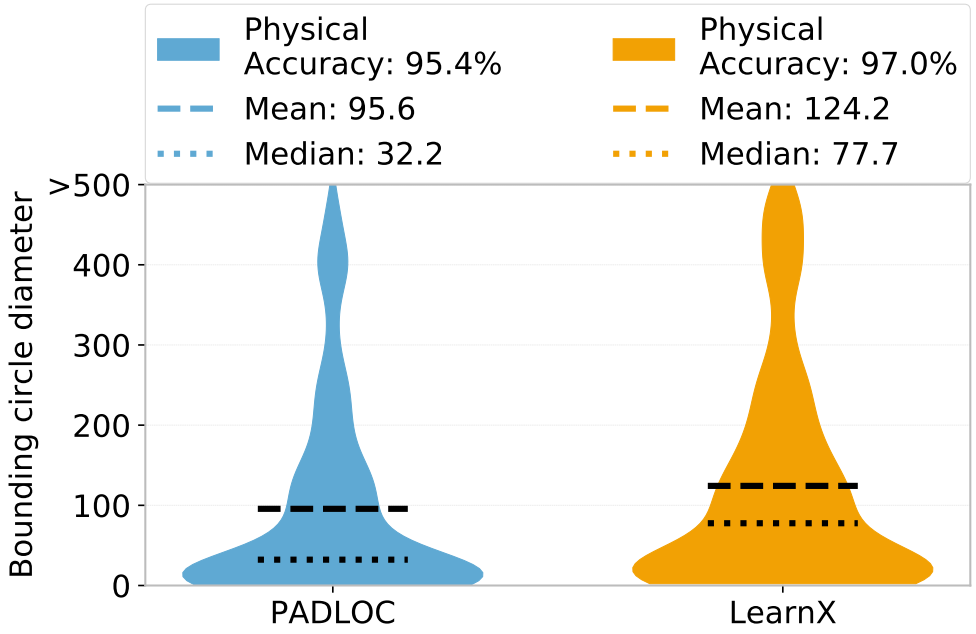


(a)

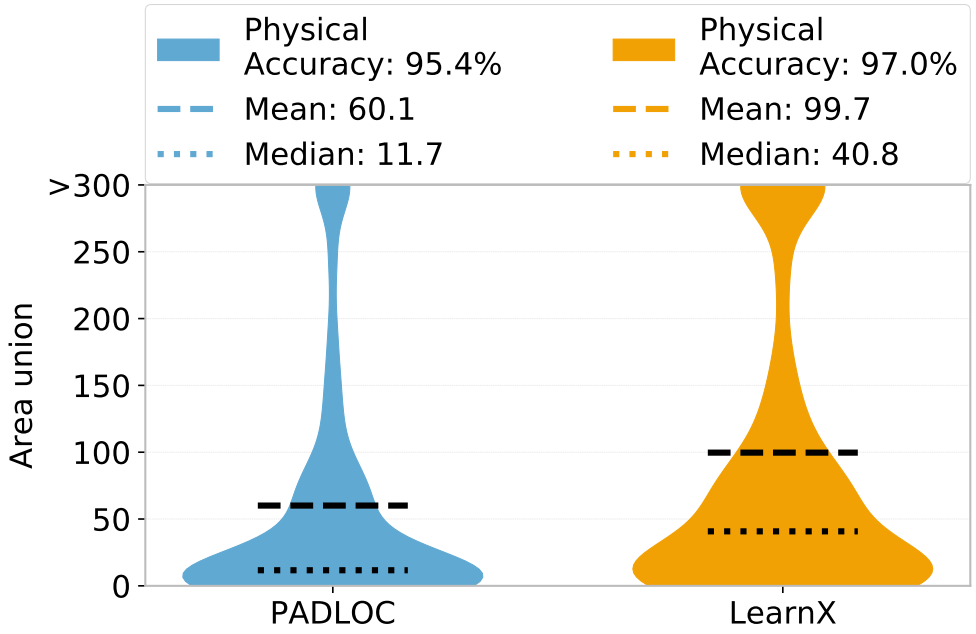


(b)

Figure 4.17: Probability density distribution of (a) *bounding circle diameter* and (b) *area union* for the design "L2B", when open defects are diagnosed by PADLOC and LearnX.



(a)



(b)

Figure 4.18: Probability density distribution of (a) *bounding circle diameter* and (b) *area union* for the design "B18", when open defects are diagnosed by PADLOC and LearnX.

4.3.2.3 Comparison with Commercial Diagnosis

In this experiment, PADLOC is compared with two state-of-the-art commercial diagnosis tools to demonstrate its potential. For each commercial diagnosis tool, the portion of the suspected design layout corresponding to a top-scoring candidate is considered as its final output.

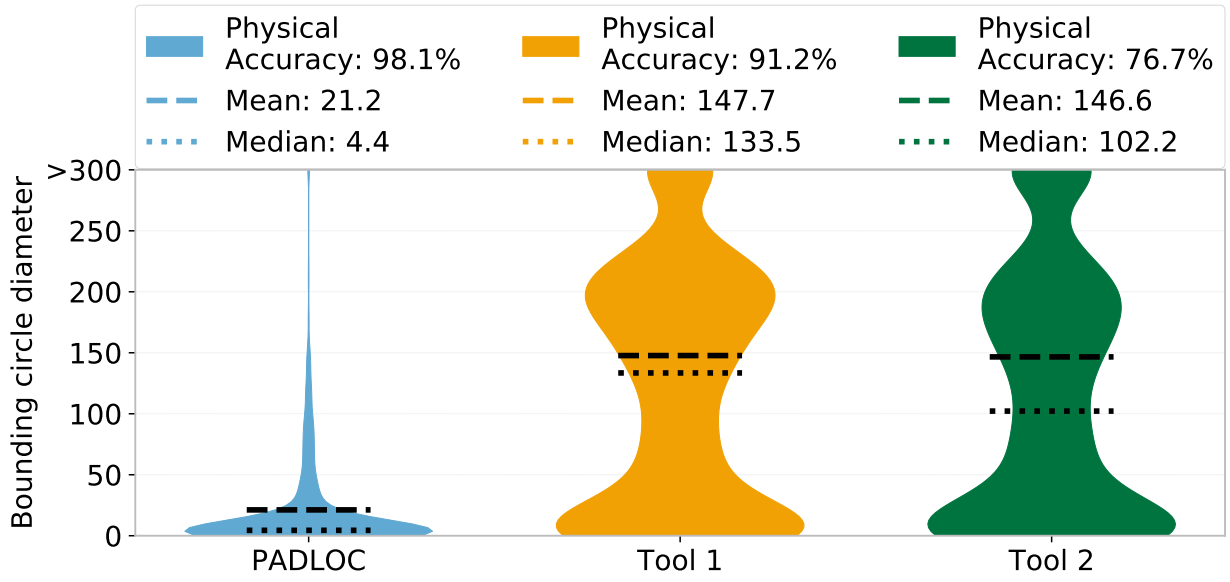
Figures 4.19 through 4.26 plot the probability density distribution of two diagnostic metrics, *bounding circle diameter* and *area union*, attained by PADLOC and commercial diagnosis. The physical accuracy, defined in Section 4.3.1, achieved by each diagnosis approach is displayed at the top of each figure. It can clearly be seen from the shapes of the distribution of *bounding circle diameter* and *area union* shown in Figures 4.19-4.26 that PADLOC achieves significantly higher physical resolution than commercial diagnosis.

Observations specific to each design regarding bridge defect diagnosis are as follows (Figures 4.19 through 4.22).

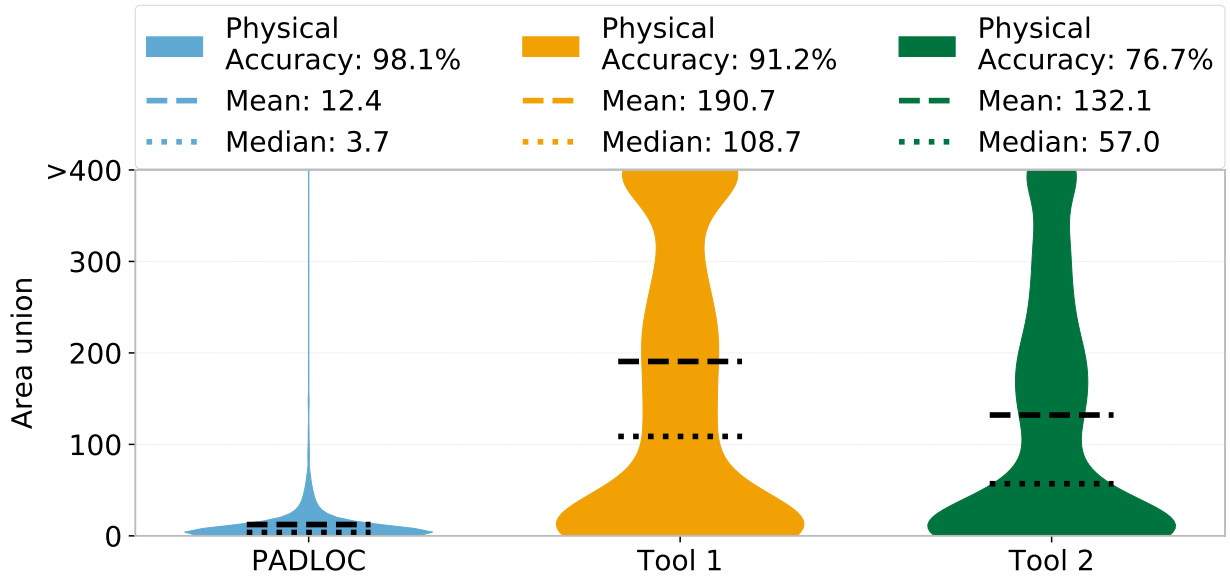
1. **AES**: (Figure 4.19) The average physical accuracy for PADLOC is 98.1%, which is 7.6% (27.9%) more than Tool 1 (Tool 2). The mean (median) diameter of the bounding circle for PADLOC is 21.2 (4.4), which is just 14.3% (3.3%) of the mean (median) *bounding circle diameter* for Tool 1, and 14.5% (4.3%) for Tool 2. In addition, PADLOC reports smaller *bounding circle diameter* than Tool 1 (Tool 2) for 64.0% (64.5%) of fail logs.

   The mean (median) *area union* for PADLOC is 12.4 (3.7), which is only 6.5% (3.4%) of the mean (median) *area union* for Tool 1 and 9.4% (6.5%) of the mean (median) *area union* for Tool 2. Moreover, PADLOC implicates smaller *area union* for 64.0% of fail logs when compared to commercial diagnosis, where PADLOC reduces *area union* by at most 9000X.

2. **DES**: (Figure 4.20) PADLOC attains an average physical accuracy of 97.1%, which is 15.6% (22.6%) better than Tool 1 (Tool 2). The mean (median) diameter of the

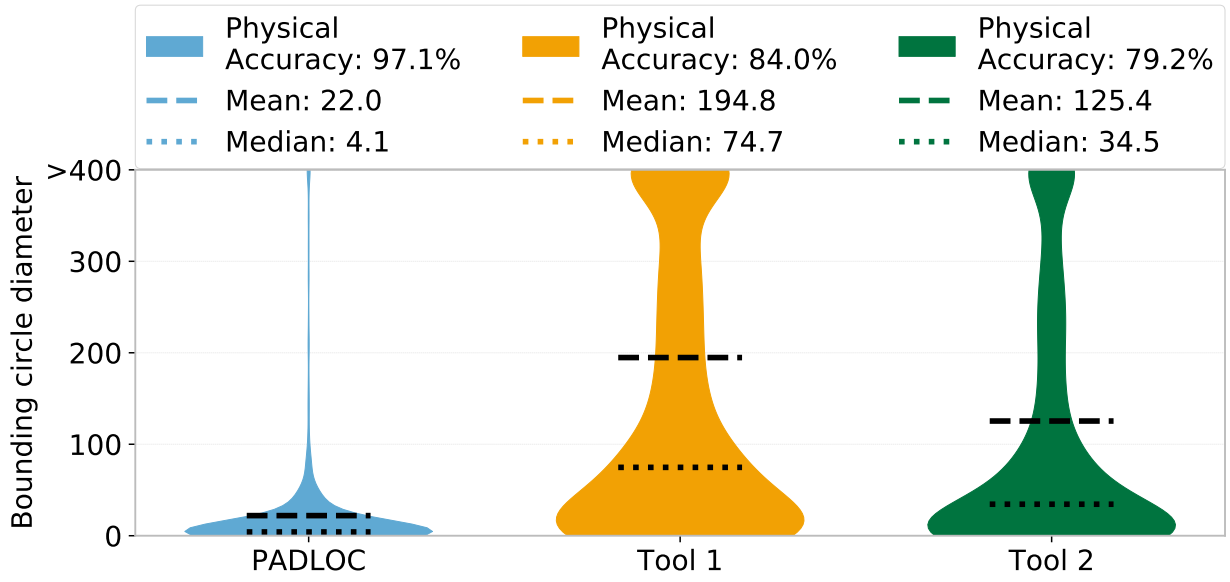


(a)

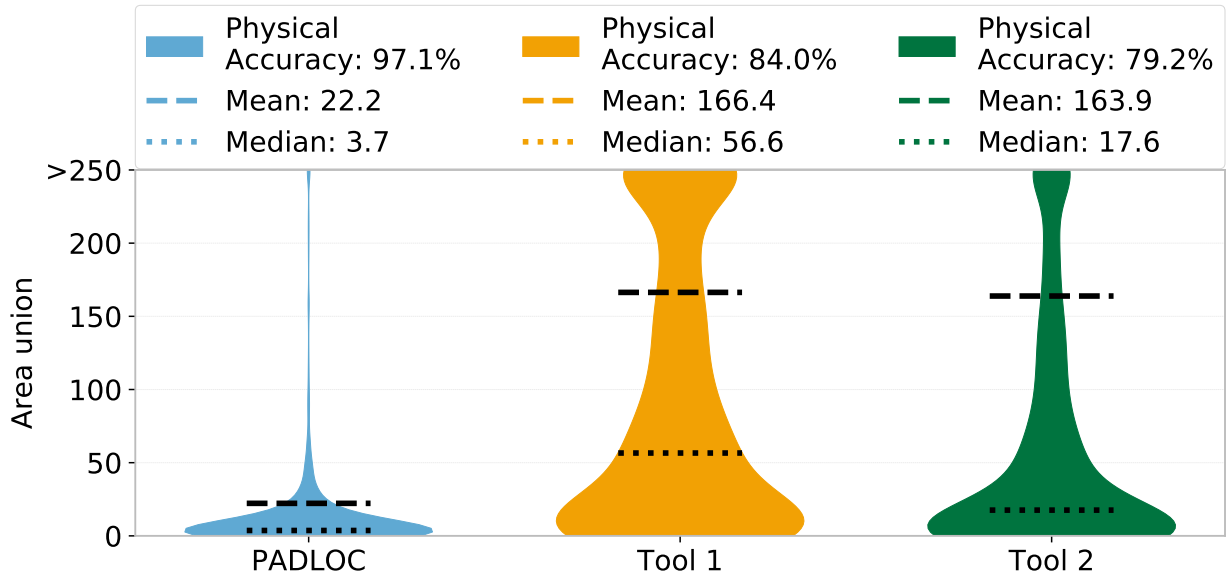


(b)

Figure 4.19: Probability density distribution of (a) *bounding circle diameter* and (b) *area union* for the design "AES", when bridge defects are diagnosed by PADLOC and commercial diagnosis.

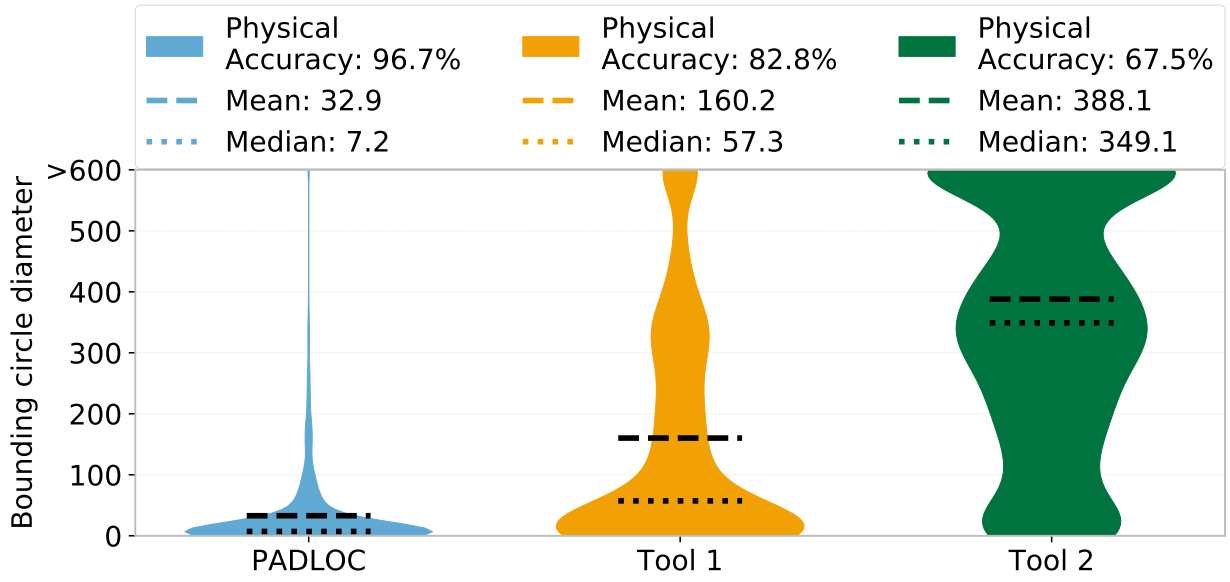


(a)

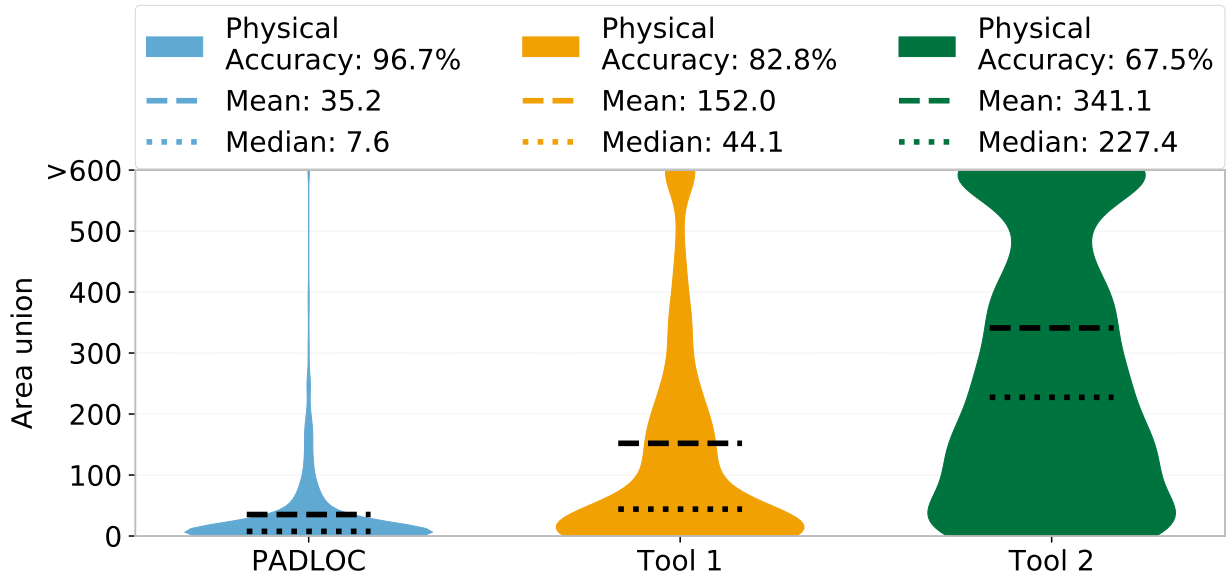


(b)

Figure 4.20: Probability density distribution of (a) *bounding circle diameter* and (b) *area union* for the design "DES", when bridge defects are diagnosed by PADLOC and commercial diagnosis.

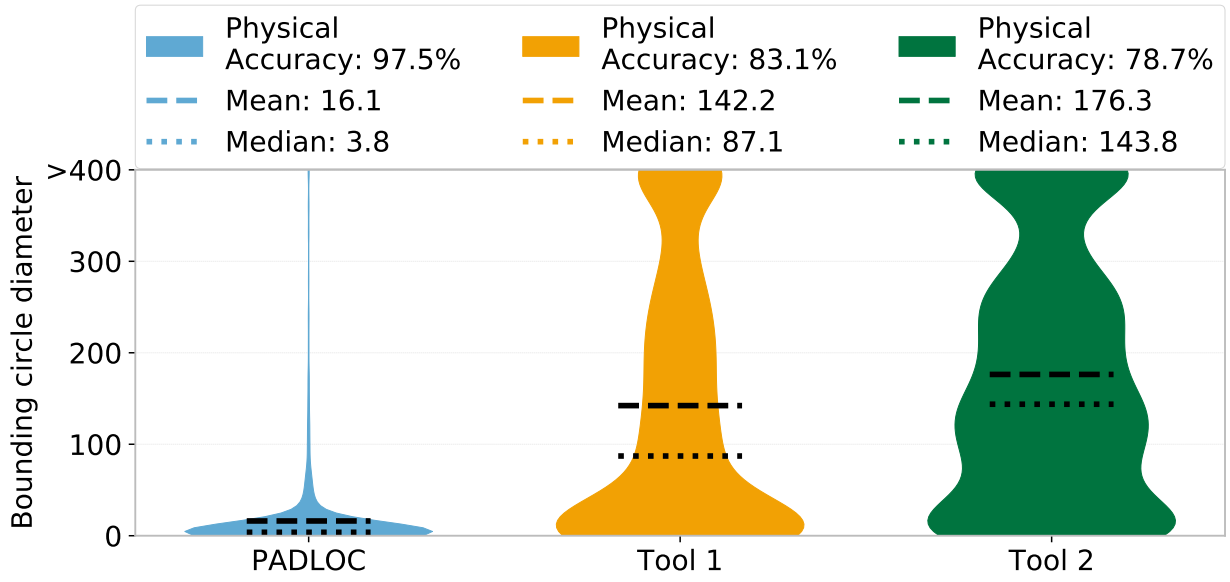


(a)

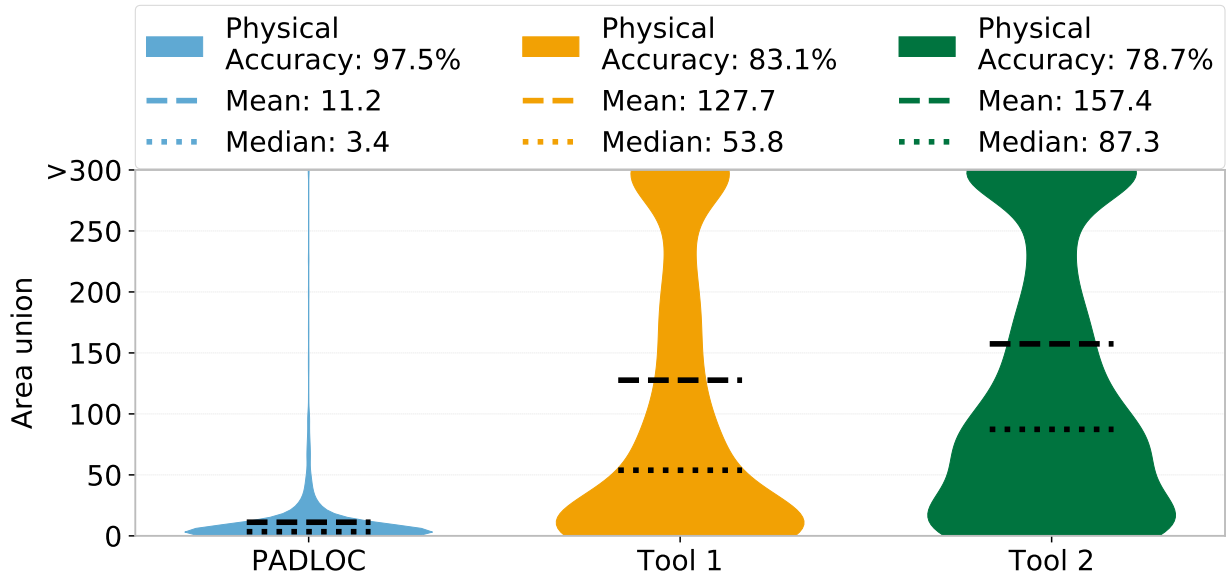


(b)

Figure 4.21: Probability density distribution of (a) *bounding circle diameter* and (b) *area union* for the design "L2B", when bridge defects are diagnosed by PADLOC and commercial diagnosis.



(a)



(b)

Figure 4.22: Probability density distribution of (a) *bounding circle diameter* and (b) *area union* for the design "B18", when bridge defects are diagnosed by PADLOC and commercial diagnosis.

bounding circle is 22.0 (4.1), which is 11.3% (5.5%) of the mean *bounding circle diameter* for Tool 1, and 17.5% (12.0%) of the mean for Tool 2. Additionally, PADLOC reports a smaller bounding circle for 63.3% (56.6%) of fail logs, in comparison with Tool 1 (Tool 2).

The mean (median) *area union* for PADLOC is 22.2 (3.7), which is 13.4% (6.5%) of the mean (median) *area union* for Tool 1 ,and 13.6% (21.0%) of the mean (median) *area union* for Tool 2. In addition, PADLOC reports smaller *area union* than Tool 1 (Tool 2) for 63.8% (55.3%) of fail logs.

3. **L2B**: (Figure 4.21) The physical accuracy achieved by PADLOC is 96.7%. PADLOC correctly diagnoses 16.8% (43.3%) more fail logs than Tool 1 (Tool 2). The mean (median) *bounding circle diameter* for PADLOC is 32.9 (7.2), whereas the mean (median) *bounding circle diameter* of Tool 1 is 4.9X (8X) and the mean (median) *bounding circle diameter* of Tool 2 is 11.8X (48.5X) times PADLOC. In addition, PADLOC reports a smaller bounding circle for 49.5% (88.8%) of the fail logs, when compared to Tool 1 (Tool 2).

   The mean *area union* reported by PADLOC is 35.2, which is approximately $1/4^{th}$ ($1/10^{th}$) of the mean for Tool 1 (Tool 2). The median *area union* for PADLOC is 7.6, which is approximately $1/6^{th}$ ($1/30^{th}$) of the median for Tool 1 (Tool 2). Moreover, PADLOC reports smaller *area union* than Tool 1 (Tool 2) for 46.3% (85.4%) of fail logs.

4. **B18**: (Figure 4.22) PADLOC identifies the defect correctly for 97.5% of fail logs, while Tool 1 (Tool 2) identifies the defect correctly for 83.1% (78.7%) of fail logs. The mean (median) *bounding circle diameter* for PADLOC is 16.1 (3.8), which is 11.3% (4.3%) of the mean *bounding circle diameter* for Tool 1, and 9.2% (2.6%) of the mean for Tool 2. Additionally, PADLOC reports a smaller bounding circle than Tool 1 (Tool 2) for

60.7% (80.7%) of fail logs.

The mean (median) *area union* for PADLOC is 11.2 (3.4), which is 8.8% (6.3%) of the mean (median) *area union* for Tool 1 and 7.1% (3.9%) of the mean (median) *area union* for Tool 2. In addition, PADLOC reports smaller *area union* than Tool 1 (Tool 2) for 60.3% (80.4%) of fail logs.

Thus, PADLOC achieves high physical accuracy (between 96.7% - 98.1%) for a failing chip affected with a bridge defect, whereas the maximum accuracy that a commercial tool can attain is 91.2%. Besides that, the enhancement in the physical resolution for PADLOC is significant. Specifically, when averaged over the four designs examined, the mean *bounding circle diameter* for PADLOC is $1/7^{th}$ ($1/9^{th}$) of the mean for Tool 1 (Tool 2). For Tool 1 (Tool 2), only 19.3% (16.1%) of fail logs have *bounding circle diameter* less than or equal to the median *bounding circle diameter* for PADLOC.

Observations specific to each design regarding open defect diagnosis are as follows (Figures 4.23 through 4.26).

1. **AES**: (Figure 4.23) PADLOC correctly locates the defect for 97.6% of fail logs, while Tool 1 (Tool 2) correctly locates the defect for 94.4% (82.3%) of fail logs. The mean (median) diameter of the bounding circle for PADLOC is 99.3 (106.5), which is $3/5^{th}$ ($2/3^{rd}$) of the mean (median) *bounding circle diameter* for Tool 1. PADLOC reports a smaller bounding circle for 63.4% (33.7%) of fail logs, in comparison with Tool 1 (Tool 2).

   The slightly improved statistical properties of physical resolution for Tool 2 (when compared to PADLOC) come at a huge cost; specifically, the accuracy for Tool 2 is 15.3% less than PADLOC.

   The mean (median) *area union* for PADLOC is 70.9 (25.2), which is 42.4% (17.9%) of the mean (median) *area union* for Tool 1 and 74.3% (58.1%) of the mean (median)

area union for Tool 2. Additionally, PADLOC reports smaller *area union* than Tool 1 (Tool 2) for 68.4% (39.6%) of fail logs.

2. **DES**: (Figure 4.24) PADLOC attains an average physical accuracy of 97.5%, which is 7.3% (29.8%) better than Tool 1 (Tool 2). The mean (median) diameter of the bounding circle is 77.0 (37.2), which is 50.8% (47.6%) of the mean (median) *bounding circle diameter* for Tool 1. Additionally, PADLOC reports a smaller bounding circle than Tool 1 for 68.0% of fail logs. The mean (median) *area union* for PADLOC is 36.1 (15.6), which is 35.1% (27.7%) of the mean (median) *area union* for Tool 1. In addition, PADLOC reports smaller *area union* than Tool 1 for 75.0% of fail logs.

   Although Tool 2 reports a smaller median *bounding circle diameter* and *area union* than PADLOC, Tool 2 is inaccurate more often. Specifically, Tool 2 inaccurately diagnoses 29.8% more fail logs than PADLOC.

3. **L2B**: (Figure 4.25) The average physical accuracy for PADLOC is 92.4%, which is 7.3% (1.3%) more than Tool 1 (Tool 2). The mean (median) diameter of the bounding circle for PADLOC is 131.8 (49.6), while the mean (median) for Tool 1 is 1.5X (2.6X) times PADLOC and the mean (median) for Tool 2 is 2.5X (6.7X) times PADLOC. In addition, PADLOC reports smaller *bounding circle diameter* than Tool 1 (Tool 2) for 45.2% (63.5%) of fail logs.

   The mean (median) *area union* for PADLOC is 143.9 (21.8), whereas the mean (median) for Tool 1 is 1.4X (1.9X) times PADLOC and the mean (median) for Tool 2 is 1.8X (5.2X) times PADLOC. Moreover, PADLOC implicates smaller *area union* for 39.3% (61.5%) of fail logs when compared to Tool 1 (Tool 2).

4. **B18**: (Figure 4.26) The physical accuracy achieved by PADLOC is 95.4%. PADLOC diagnoses 8.8% (3.0%) more fail logs correctly than Tool 1 (Tool 2). The mean (median)

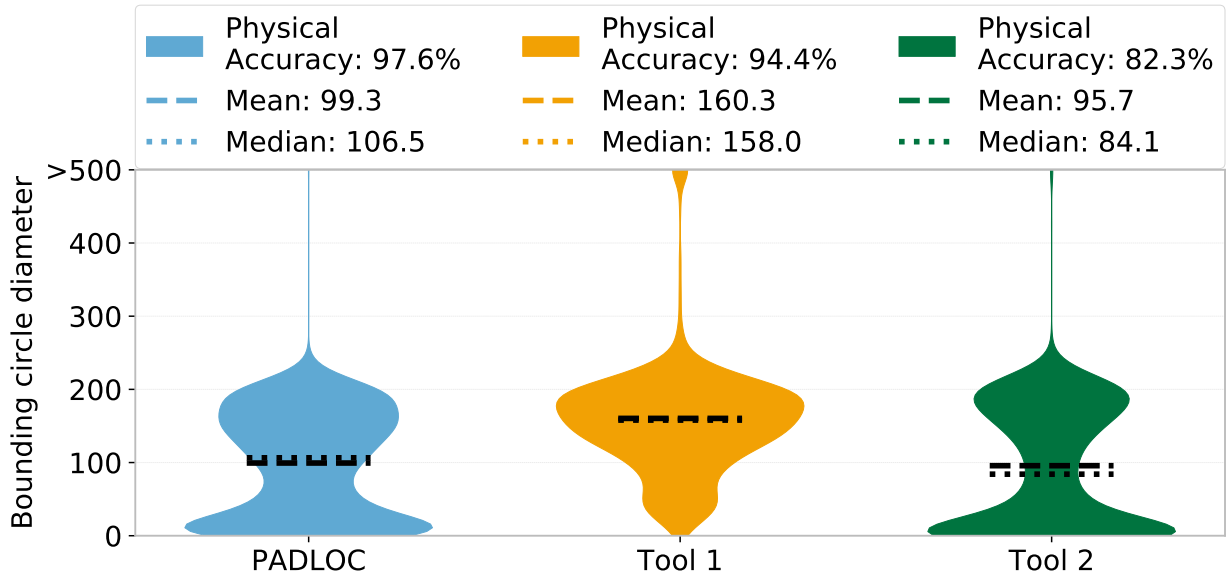
bounding circle diameter for PADLOC is 95.6 (32.2), whereas the mean (median) *bounding circle diameter* of Tool 1 is 1.6X (3.6X) and the mean (median) *bounding circle diameter* of Tool 2 is 1.6X (3.7X) times PADLOC. In addition, PADLOC reports smaller *bounding circle diameter* than Tool 1 (Tool 2) for 62.1% (64.7%) of fail logs.

The mean *area union* for PADLOC is 60.1, which is 42.8% (50.1%) of the mean for Tool 1 (Tool 2). The median *area union* for PADLOC is 11.7, which is approximately $1/5^{th}$ ($1/6^{th}$) of the median for Tool 1 (Tool 2). Moreover, PADLOC reports smaller *area union* than Tool 1 (Tool 2) for 66.7% (69.7%) of fail logs.

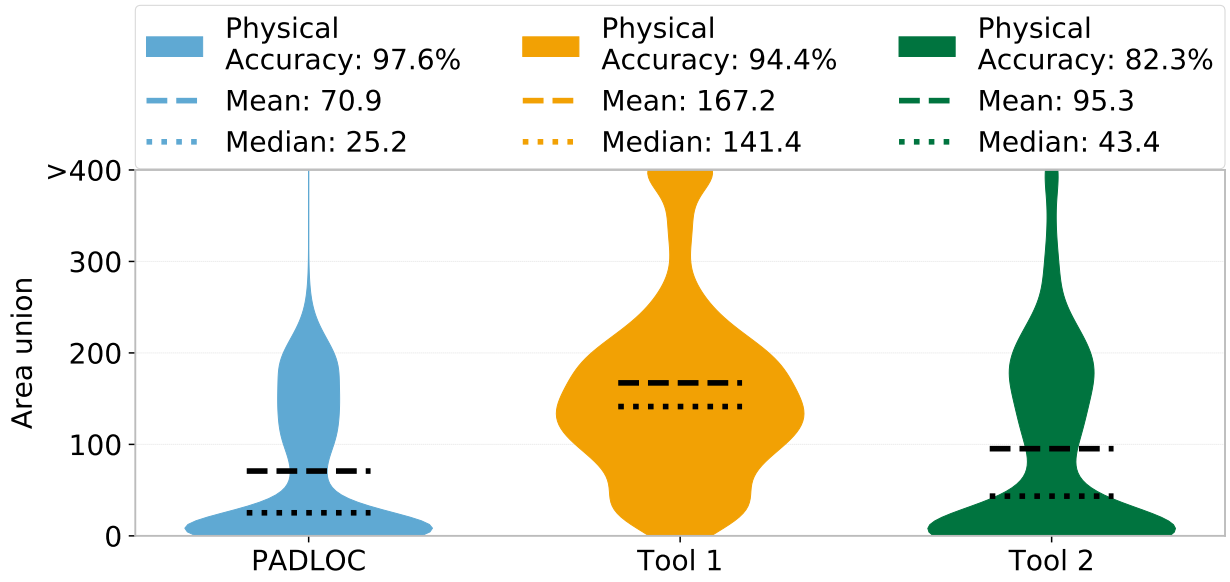
Thus, PADLOC achieves high physical accuracy (between 92.4% - 97.6%) for a failing chip affected with an open defect, where the minimum and maximum improvement over a commercial tool is 1.3% and 29.8%, respectively. Besides that, the enhancement in the physical resolution for PADLOC is remarkable. Specifically, when averaged over the four designs analyzed, the mean *bounding circle diameter* for PADLOC is 60.8% (62.8%) less than the mean for Tool 1 (Tool 2). For Tool 1 (Tool 2), only 26.0% (39.3%) of the fail logs have *bounding circle diameter* less than or equal to the median *bounding circle diameter* attained by PADLOC.

Two defect mechanisms commonly encountered in advanced process nodes are via misalignment and high via resistance [364]. When defects due to a misaligned via are diagnosed, it is observed that the physical accuracy achieved by PADLOC is 96.3%, which is 11.2% (22.6%) more than Tool 1 (Tool 2). The mean *bounding circle diameter* for PADLOC is 25.1, which is 6.6X (8.7X) times smaller than Tool 1 (Tool 2). The mean *area union* for PADLOC is 23.0, which is 7.3X (9.1X) times smaller than Tool 1 (Tool 2).

When defects due to high via resistance are analyzed, it is revealed that PADLOC pinpoints 94.9% of defects correctly, which is 5.3% (12.8%) more than Tool 1 (Tool 2). The mean *bounding circle diameter* for PADLOC is 101.7, which is 1.6X times smaller than both

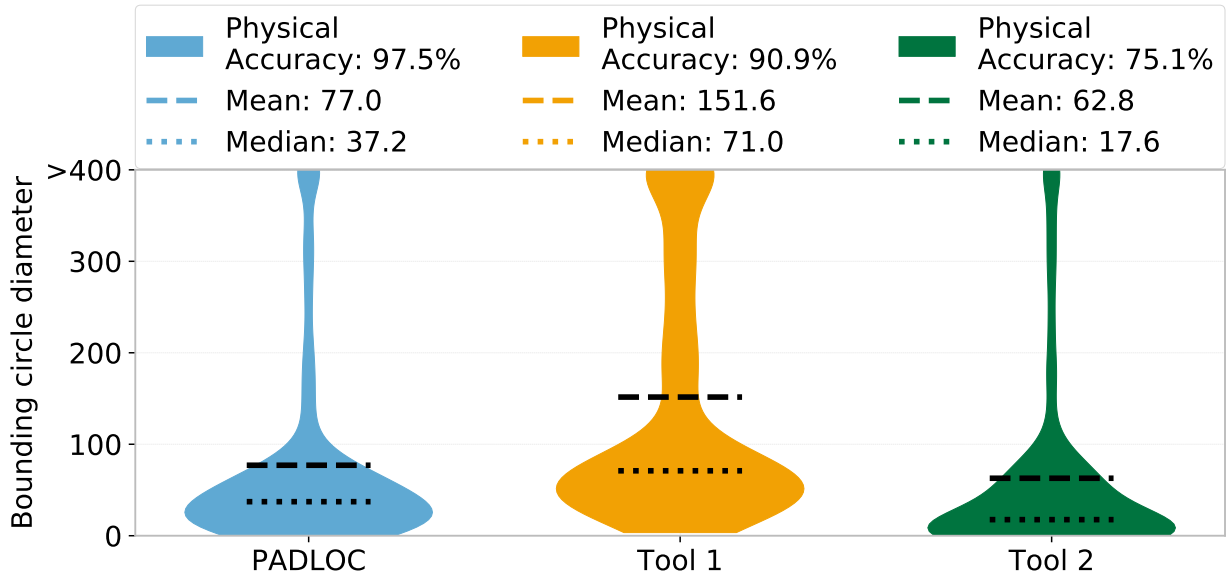


(a)

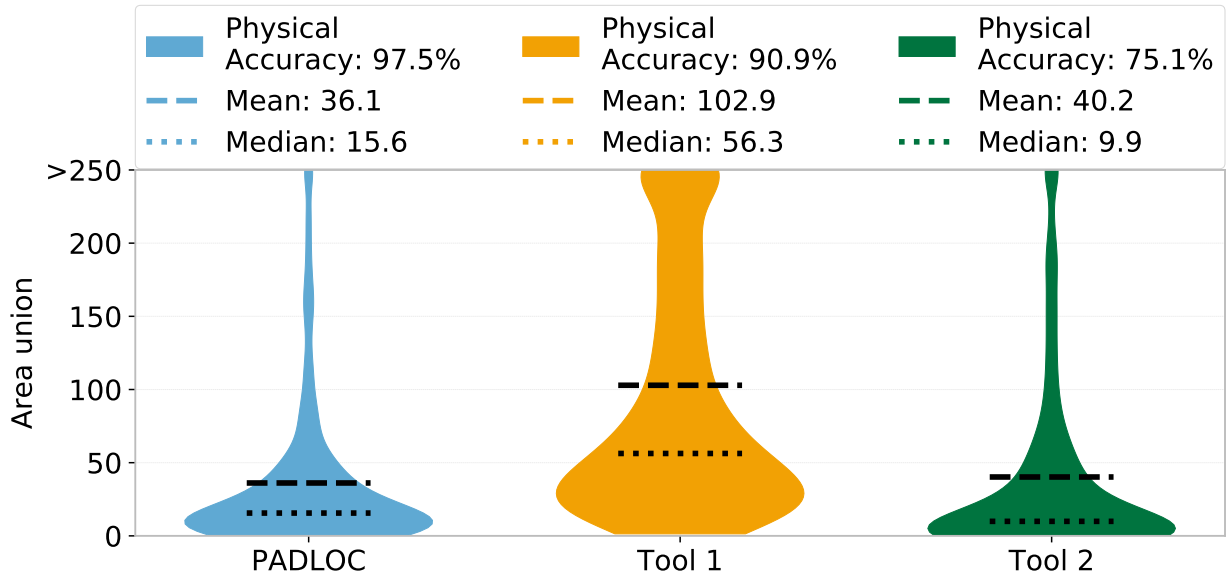


(b)

Figure 4.23: Probability density distribution of (a) *bounding circle diameter* and (b) *area union* for the design "AES", when open defects are diagnosed by PADLOC and commercial diagnosis.

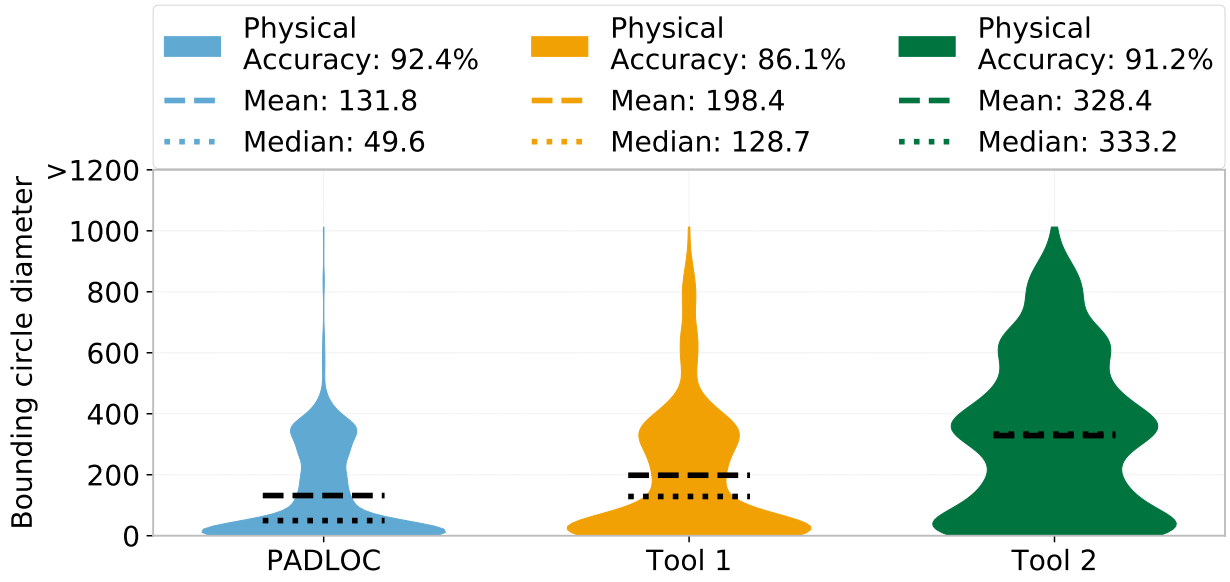


(a)

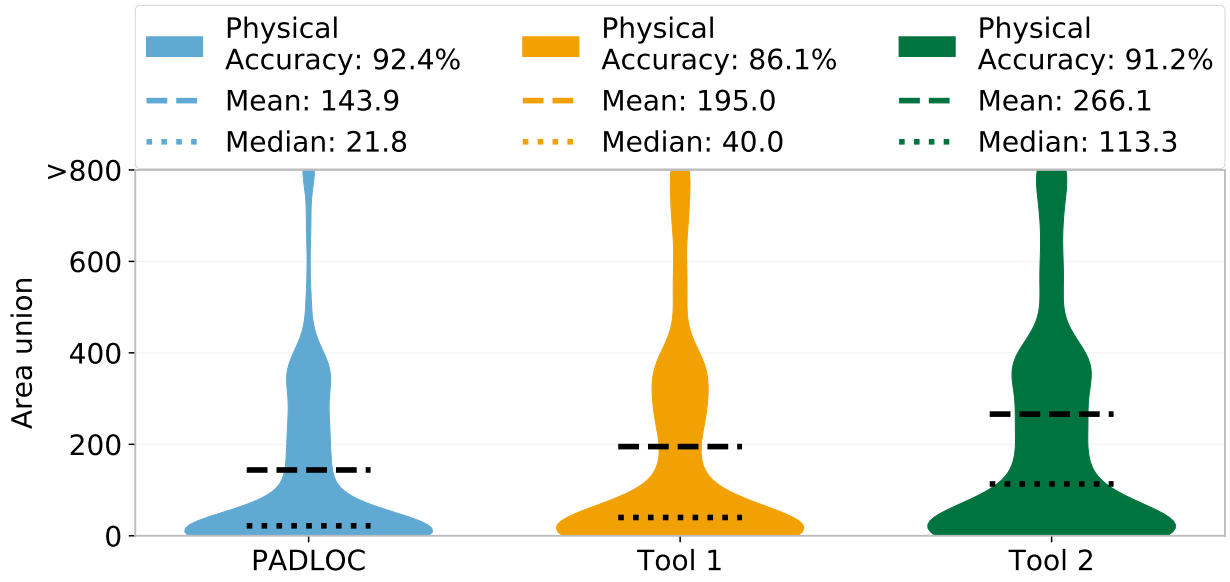


(b)

Figure 4.24: Probability density distribution of (a) *bounding circle diameter* and (b) *area union* for the design "DES", when open defects are diagnosed by PADLOC and commercial diagnosis.

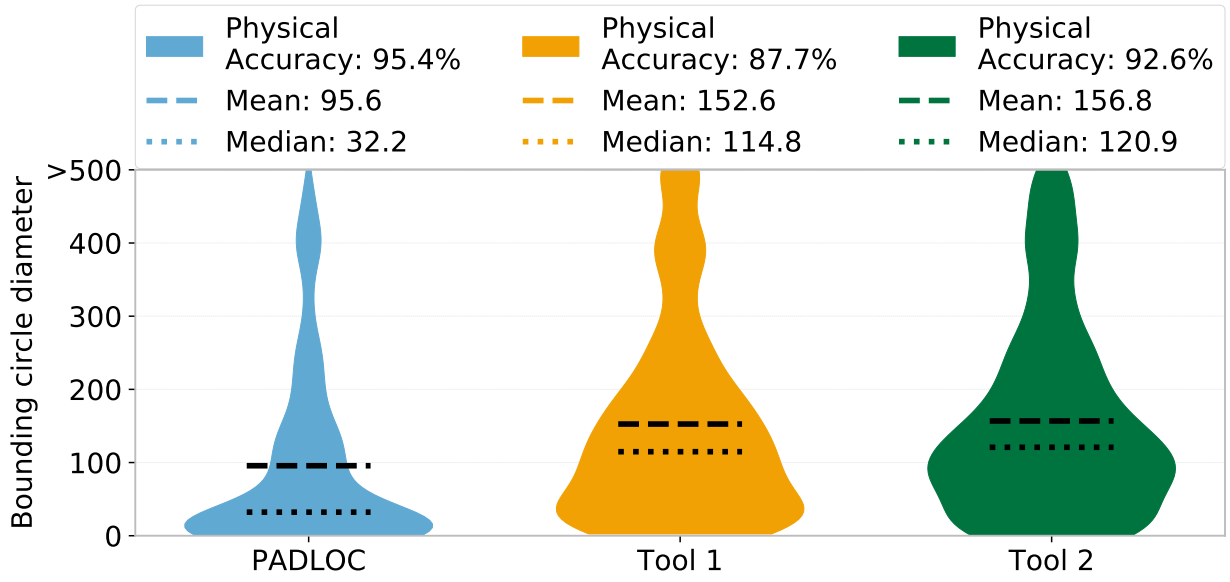


(a)

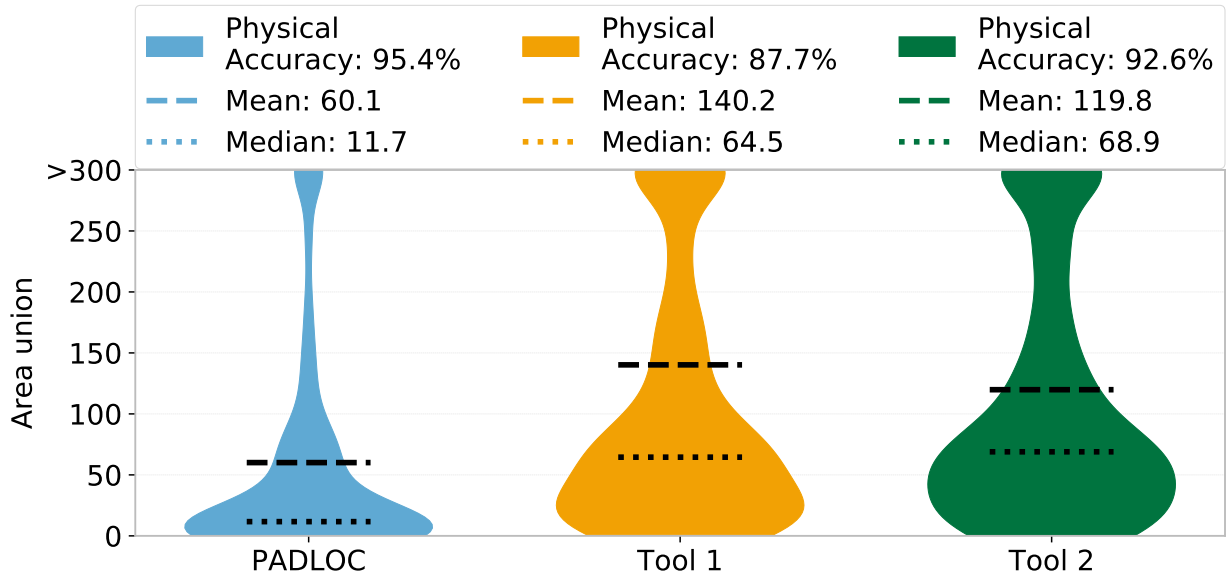


(b)

Figure 4.25: Probability density distribution of (a) *bounding circle diameter* and (b) *area union* for the design "L2B", when open defects are diagnosed by PADLOC and commercial diagnosis.



(a)



(b)

Figure 4.26: Probability density distribution of (a) *bounding circle diameter* and (b) *area union* for the design "B18", when open defects are diagnosed by PADLOC and commercial diagnosis.

the commercial tools. The mean *area union* for PADLOC is 82.8, which is 1.7X times smaller than Tool 1 and Tool 2.

4.3.2.4 Results Summary

Experiment results presented in Section 4.3.2.2 demonstrate the advantages of PADLOC, when compared to LearnX. It is revealed that PADLOC correctly finds the defect location for 96.5% of fail logs, when averaged over different designs and defect types. With a little impact on physical accuracy when compared to LearnX (specifically, 0.8%), the enhancement in the physical resolution (quantified using two metrics, *bounding circle diameter* and *area union*) for PADLOC is appreciable. The mean (median) *bounding circle diameter* for PADLOC is 15.6% (28.7%) less than LearnX. PADLOC reports a smaller bounding circle for 20.1% of fail logs, while reducing *bounding circle diameter* by at most 300X. In addition, the mean (median) *area union* for PADLOC is 26.4% (37.0%) better than LearnX. PADLOC implicates smaller *area union* for 22.6% of fail logs, and decreases *area union* by up to a factor of 625.

In Section 4.3.2.3, PADLOC is pitted against state-of-the-art commercial diagnosis to investigate its potential. It is revealed that PADLOC prevails over commercial diagnosis. Figure 4.27 highlights the improvement in the physical accuracy (in comparison to leading-edge commercial diagnosis) obtained by PADLOC for the four designs examined. The horizontal axis shows the accuracy attained by each diagnosis approach for each design. The vertical axis represents each design. The increase in the percentage of fail logs that are correctly diagnosed is shown in the figure besides each plot-bar corresponding to commercial diagnosis.

It is observed from Figure 4.27 that the maximum improvement over Tool 1 is produced for B18 (specifically, 13.0%), and over Tool 2 for DES (specifically, 26.0%). The accuracy for PADLOC is at least 95.5% (for L2B) and at most 98.5% (for DES). On the other hand, the accuracy for Tool 1 ranges from 84.5% to 92.8%, and Tool 2 ranges from 77.2% to 85.7%.

Figure 4.28 and Figure 4.29 illustrate the mean *bounding circle diameter* and *area union*, respectively, reported by PADLOC and commercial diagnosis for the four designs analyzed. Because a smaller value of each physical resolution metric is preferable, the inverse of each

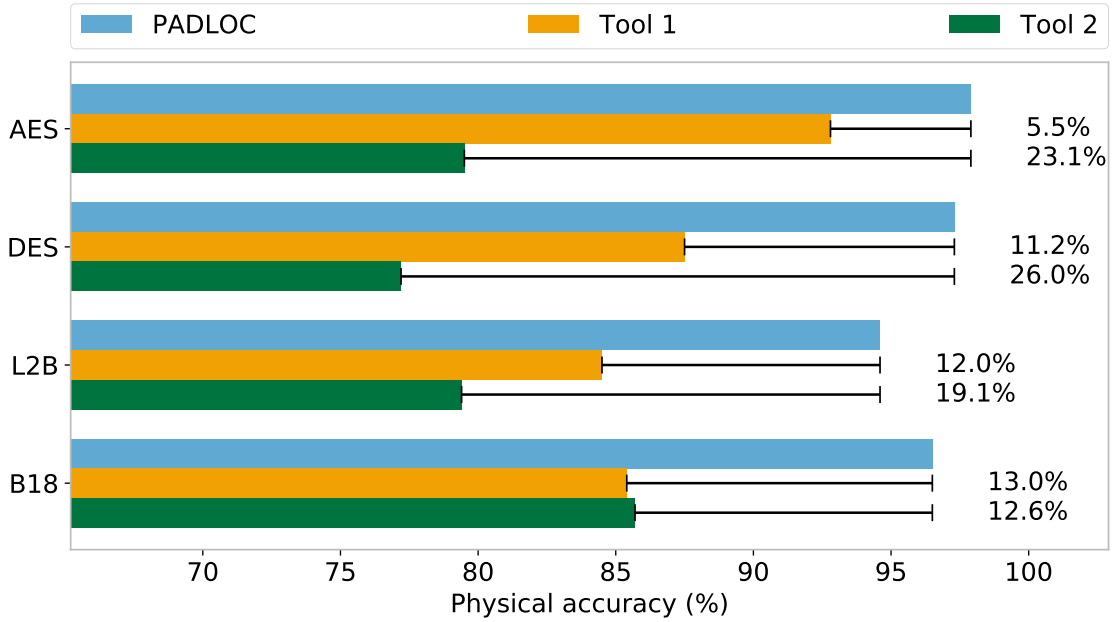


Figure 4.27: Physical accuracy achieved by PADLOC and commercial diagnosis.

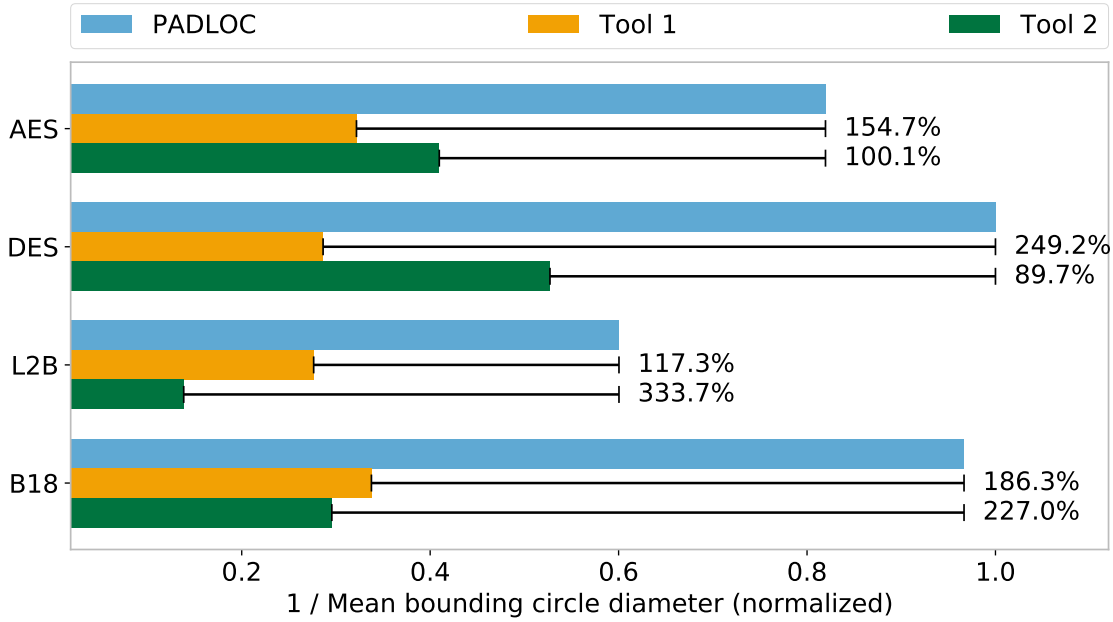


Figure 4.28: *Bounding circle diameter* achieved by PADLOC and commercial diagnosis.

metric is plotted to better visualize the effectiveness of PADLOC. Thus, in Figures 4.28 and 4.29, a longer plot-bar implies better physical resolution.

It is observed from Figure 4.28 that the improvement in *bounding circle diameter* over Tool 1 (Tool 2) is maximum for DES (L2B). The mean *bounding circle diameter* for PADLOC is at least 2.2X times (for L2B) and at most 3.5X times (for DES) better than Tool 1. The mean *bounding circle diameter* for PADLOC is at least 89.7% better than (for DES) and at most 4.4X times (for L2B) smaller than Tool 2.

Figure 4.29 reveals that the enhancement in *area union* over Tool 1 (Tool 2) is maximum for DES (B18). The average *area union* for PADLOC is at least 93.3% better than (for L2B) and at most 4.6X times (for DES) smaller than Tool 1. The mean *area union* for PADLOC is at least 2.7X (for AES) and at most 4.3X times (for B18) better than Tool 2.

When averaged over all the four designs, PADLOC achieves a physical accuracy of 96.5%, which is 10.3% (20.1%) better than Tool 1 (Tool 2). In addition, the physical resolution for

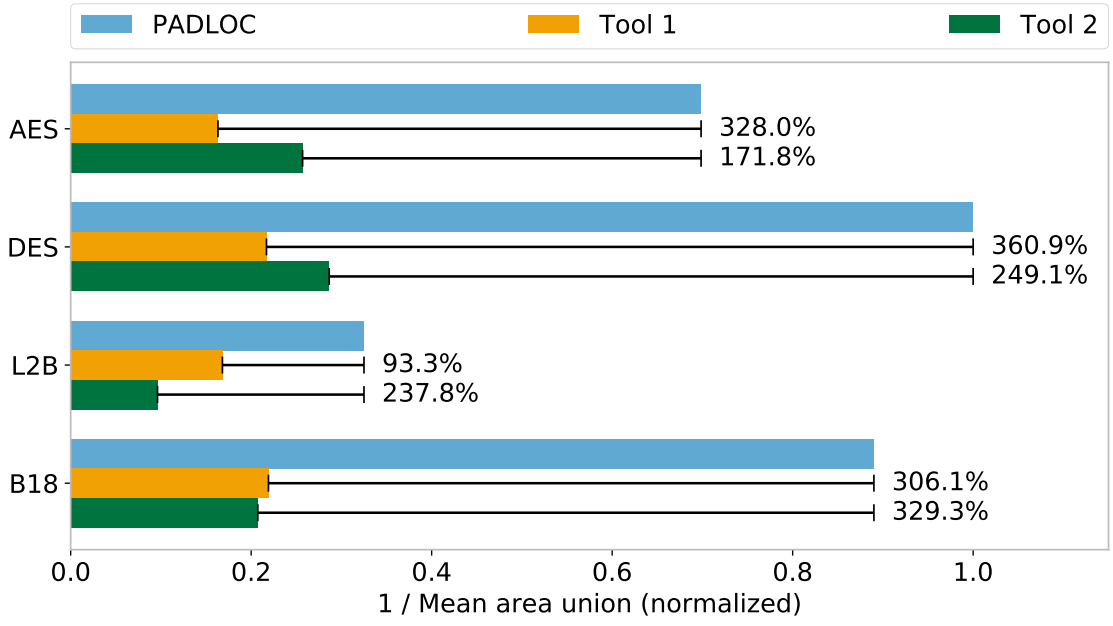


Figure 4.29: *Area union* achieved by PADLOC and commercial diagnosis.

PADLOC is impressive. The mean (median) *bounding circle diameter* for PADLOC is 61.2 (13.2), and is 37.4% (12.1%) of the mean (median) for Tool 1 and 33.0% (11.5%) of the mean (median) for Tool 2. PADLOC reports smaller *bounding circle diameter* than Tool 1 (Tool 2) for 59.4% (60.6%) of fail logs. For Tool 1 (Tool 2), only 18.5% (24.4%) of the fail logs have *bounding circle diameter* less than or equal to the median *bounding circle diameter* for PADLOC. The mean (median) *area union* for PADLOC is 48.8 (7.8), and is $1/3^{rd}$ ($1/11^{th}$) of the mean (median) for Tool 1 and $1/3^{rd}$ ($1/13^{th}$) of the mean (median) for Tool 2. PADLOC reports smaller *area union* than Tool 1 (Tool 2) for 60.3% (61.7%) of fail logs.

4.3.3 Silicon Experiment

In addition to the simulation experiment presented in Section 4.3.2, the value of PADLOC is demonstrated using an industrial design fabricated in an advanced process node. The setup for the conducted experiment is described in Section 4.3.3.1. It conveys basic information about the chip and the available silicon failure data that is used to study the effectiveness of PADLOC. The results of the experiment are discussed in Section 4.3.3.2 to explore how PADLOC fares in the real world.

4.3.3.1 Setup

PADLOC is assessed using an industrial chip that is manufactured using a 14nm process. The chip, that measures 12 mm^2 in size, is divided into 12 partitions, where each partition consists of approximately 3.5 million gates. Each partition is tested using approximately 825 test patterns that have been generated using a commercial ATPG software. Fail logs corresponding to 2,400 failing chips, of which 36 have been PFA'ed, are analyzed in this experiment. The PADLOC flow overviewed in Figure 4.6 is applied to each fail log. Each fail log is also examined by commercial diagnosis[7], where only the top-scoring candidates

[7]The available data conforms to only one of the commercial diagnosis tools (Tool 1).

are considered as its output. Each diagnosis technique is gauged using the diagnosis metrics defined in Section 4.3.1. The physical accuracy for each diagnosis method is compared for the 36 chips for which PFA results are available, while the physical resolution is calculated for all the 2,400 failing chips.

4.3.3.2 Results

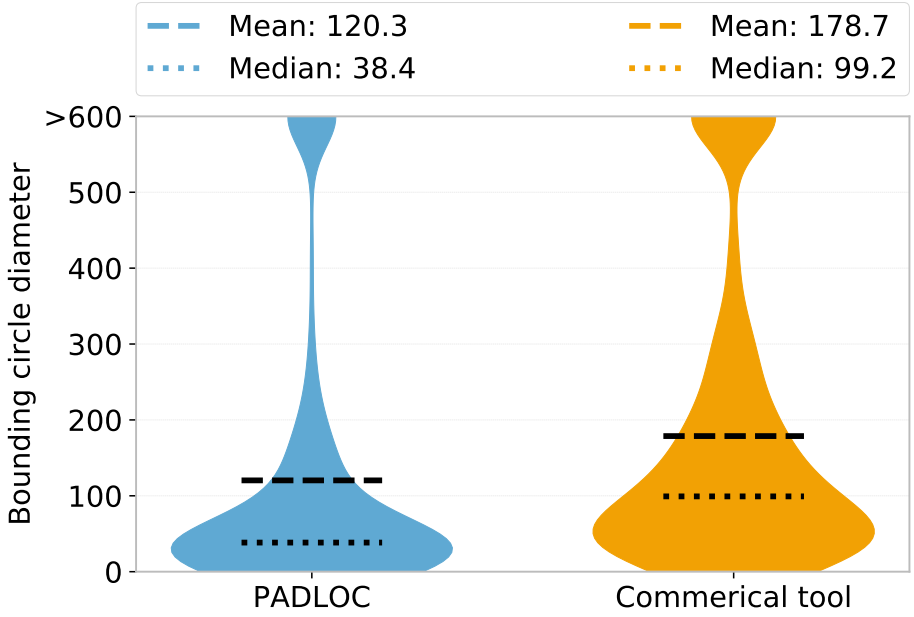
Figure 4.30 shows the probability density distribution of two diagnostic metrics, *bounding circle diameter* and *area union*, computed for PADLOC and commercial diagnosis.

Figure 4.30(a) reveals that the mean *bounding circle diameter* for PADLOC is 120.3, an improvement of 32.7% over commercial diagnosis. The median *bounding circle diameter* for PADLOC is 38.4, which is less than one third of the median computed for commercial diagnosis. In addition, PADLOC reports a smaller enclosing circle for 68.8% of fail logs, and attains 5X improvement, on average.

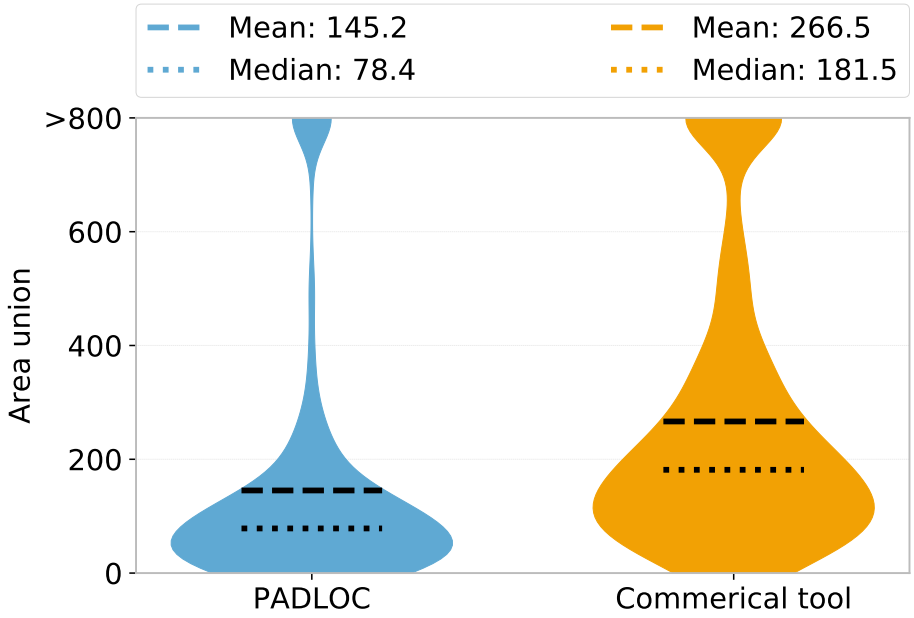
It is seen from Figure 4.30(b) that the mean *area union* for PADLOC is 145.2, while the mean *area union* for commercial diagnosis is 83.5% more than that. Additionally, the median *area union* for PADLOC is less than half of the median computed for commercial diagnosis. Furthermore, PADLOC reports smaller *area union* for 70.3% of fail logs, and achieves 9X enhancement, on average.

Among 2,400 failing chips, 36 failing chips are PFA'ed. **PFA confirms that PADLOC correctly pinpoints the correct candidate for each failing chip.** More importantly, PADLOC attains superior physical resolution without sacrificing accuracy. Figures 4.31 and 4.32 plot *bounding circle diameter* and *area union*, respectively, for each failing chip for which PFA results are available. The index of each fail log is shown on the horizontal axis, while the vertical axis shows the logarithmic value of the diagnostic metric for each fail log.

Figure 4.31 shows that the median *bounding circle diameter* for PADLOC is 134.9, which is 25.3% better than commercial diagnosis. PADLOC improves *bounding circle diameter* for



(a)



(b)

Figure 4.30: Probability density distribution of (a) *bounding circle diameter* and (b) *area union* when 2,400 silicon failures are analyzed by PADLOC and commercial diagnosis.

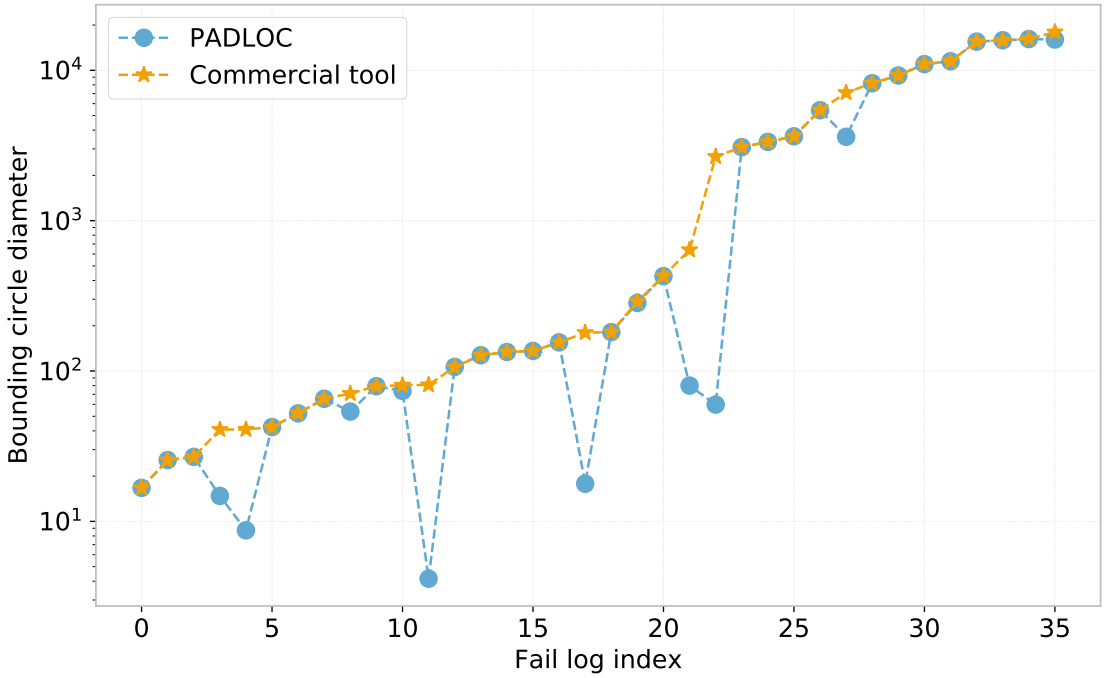


Figure 4.31: *Bounding circle diameter* for 36 failing chips that are PFA'ed.

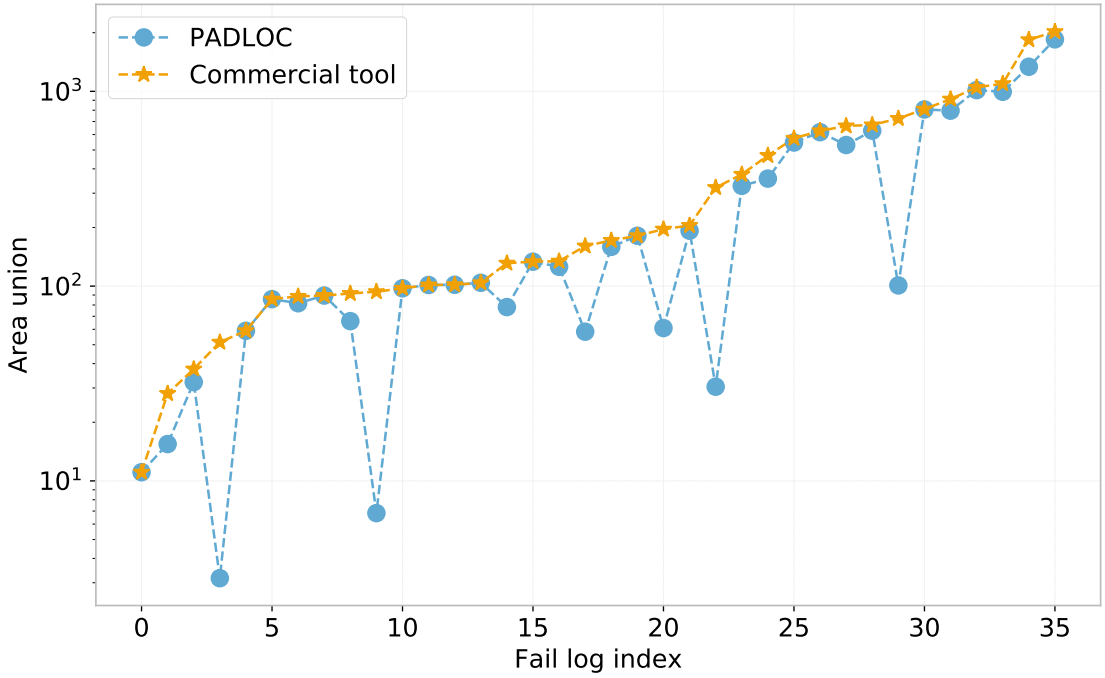


Figure 4.32: *Area union* for 36 failing chips that are PFA'ed.

17 (47.2%) fail logs. Among the fail logs for which PADLOC reports smaller *bounding circle diameter*, PADLOC achieves 6X improvement, on average, and attains at most 44X improvement over commercial diagnosis.

Figure 4.32 indicates that the median *area union* for PADLOC is 102.8, which is 38.1% better than commercial diagnosis. PADLOC reports a smaller value of *area union* for 26 (72.2%) fail logs. Among these 26 fail logs, PADLOC attains 3X average improvement, and achieves at most 16X improvement over commercial diagnosis.

The biggest takeaway here is that PADLOC is 100.0% accurate in localizing a defect in each failing chip, while reporting significantly better physical resolution than commercial diagnosis.

4.4 Conclusion

This chapter describes a comprehensive physically-aware methodology for defect diagnosis we call PADLOC (Physically-Aware Defect LOcalization and Characterization). PADLOC focuses on physical localization and behavior derivation of a back-end defect (i.e., a defect that resides outside a standard cell). Specifically, it identifies the precise $x-y-z$ location of a defect using minimal layout information. It analyzes the topology of the net(s) involved in a defect and its physical neighborhood to pinpoint the subnets that are likely to correspond to a defect. It avoids the use of possibly unreliable extracted parasitics to derive the logic value at a defect location. Instead, it deduces the behavior of a defect based on its physical and logical neighborhood.

A thorough simulation-based experiment is designed to assess the performance of PADLOC. Results from simulation experiments for 8,000 defects reveal that PADLOC achieves an accuracy of 96.5%, which is 10.3% and 20.1% better than two state-of-the-art commercial diagnosis tools. More impressive, however, is the physical resolution attained by PADLOC.

Two metrics, *bounding circle diameter*, which is defined as the diameter of the smallest circle enclosing the layout regions suspected by diagnosis, and *area union*, which is defined as the area of the geometric union of the implicated layout regions, are employed in this work to quantify physical resolution. It is observed that the average *bounding circle diameter* for PADLOC is 61.2, and is 33.0% of the mean computed for the better-of-the two commercial tools. The average *area union* for PADLOC is 48.8, which is at least 29.5% of the mean *area union* for commercial diagnosis.

A silicon experiment is also conducted to demonstrate the capability of PADLOC. Silicon failure data corresponding to 2,400 failing chips fabricated in an advanced process technology is examined. The analysis reveals that PADLOC reports smaller *bounding circle diameter* as well as *area union* for 68.8% and 70.3% of fail logs, respectively, when compared with commercial diagnosis. In addition, the *bounding circle diameter* for PADLOC is 120.3, on average, which is 32.7% smaller (i.e., better) than commercial diagnosis.

PADLOC is further validated by inspecting 36 failing chips that are PFA'ed. PADLOC is able to physically locate a defect in each failing chip correctly, while suspecting a smaller layout region than state-of-the-art commercial diagnosis. It is seen that PADLOC improves *bounding circle diameter* for 47.2% of fail logs, and achieves at most 44X improvement.

Locating a defect correctly while implicating a smaller layout region guides a volume diagnosis approach effectively to identify the failing root cause responsible for yield loss. Improved physical localization likely increases the percentage of failing chips that are suitable for PFA. This is because more failing chips would have their suspected defective areas less than the threshold determined by PFA [161]. Thus, a higher percentage of failing chips can be PFA'ed efficiently, which translates to decreased PFA cost. Additionally, enhanced defect localization accelerates PFA because a smaller layout region needs to be searched for a defect, which in turn, likely, facilitates rapid yield learning.

Chapter 5

NOIDA: Noise-resistant Intra-cell Diagnosis

Chapter 2 presents LearnX, a physically-aware diagnosis methodology that concentrates on locating a defect in two phases. A defect that mirrors the behavior of a common fault model such as the stuck-at and the wired bridge fault model is identified in Phase 1. A defect that exhibits non-trivial behavior is diagnosed in Phase 2 using a machine learning model that learns the hidden correlations between a correct candidate and the observed circuit response.

While LearnX focuses on localizing a single defect affecting a failing chip, Chapter 3 discusses MD-LearnX that focuses on localizing multiple defects. MD-LearnX is a three-phase diagnosis methodology. In MD-LearnX, Phases 1 and 2 identify non-interacting defects. Phase 3, on the other hand, is proficient in identifying multiple, interacting defects.

Chapter 4 introduces PADLOC, which acts upon the output of LearnX/MD-LearnX to physically localize and derive the logic behavior of a defect. It enhances the physical resolution for a defect by partitioning a candidate into segments based on its topology and the topology of its neighborhood. It determines the precise behavior of a defect by investigating the logical activity of its surrounding circuitry.

However, PADLOC is specifically developed for inspecting a back-end defect, i.e., a defect that affects one or more interconnects in a design. With advanced technology nodes, there has been an increasing number of front-end (i.e., within a standard cell) defects. Conventional diagnosis approaches typically fail to localize such defects. In addition, circuit-level noise can change the tester response in an unexpected way, and can decrease the quality of diagnosis. This work attempts to address these concerns. That is, a noise-resistant diagnosis methodology, called NOIDA (NOise-resistant Intra-cell Diagnosis Approach), is developed to effectively diagnose a front-end defect. NOIDA analyzes the intra-cell physical neighborhoods surrounding likely defect locations to locate an intra-cell defect. Defect behavior is derived based on the neighborhood, instead of querying a pre-computed fault dictionary. Thus, while PADLOC focuses on identifying the physical location and the logic behavior of a back-end defect, NOIDA centers on executing the same for a front-end defect.

The rest of the chapter is organized as follows. Section 5.1 motivates the need for a front-end defect-diagnosis methodology. Section 5.2 discusses prior work related to the physical localization of a front-end defect. It also motivates NOIDA and reasons the need to design a diagnosis approach that is robust to circuit-level noise. The details of how NOIDA derives the behavior and pinpoints the physical location of a defect within a cell are described in Section 5.3. Results from a comprehensive simulation-based experiment are presented in Section 5.4, where NOIDA is shown to be more effective in diagnosing a defect than state-of-the-art commercial diagnosis. Finally, Section 5.5 concludes this chapter by summarizing the overall contributions of this work.

5.1 Motivation

Diagnosis methods reviewed and discussed in Section 2.1, Section 3.1 and Section 4.1 focus on localizing (and sometimes, characterizing) a back-end defect, and by extension, differenti-

ating between a back-end and a front-end defect. That is, such a diagnosis method is unable to pinpoint the location of a defect within a standard cell, and reports either an interconnect, a standard-cell pin or the cell itself as a possible candidate.

However, identifying the location of a defect within a failing standard cell is equally important due to the following reasons.

1. Standard-cell failing frequency

Decreasing feature sizes and increasing complexity of the manufacturing process has led to an increased number of front-end defects. Several PFA results, based on technologies ranging from 160nm to 14nm, have been disclosed in the literature where a front-end diagnosis approach assisted in identifying the root cause of the failure within a cell [74, 175–179].

To further demonstrate the importance of front-end diagnosis, silicon test data from thousands of failing chips manufactured by various organizations in process nodes ranging from 55nm to 14nm is collected and analyzed. The results are summarized in Table 5.1. The first column ("Chip type") specifies whether silicon data is obtained for a test chip or high-volume manufacturing chip. The second column shows the process node in which a chip is fabricated.

Each fail log obtained is diagnosed using a commercial diagnosis software. The third column shows the percentage of chips where the top-scoring candidate is uniquely classified as a cell candidate. Thus, the values shown in this column (likely) guarantee the existence of a front-end defect. The fourth column, on the other hand, shows the percentage of chips where one of the top-scoring candidates reported by commercial diagnosis is categorized as a cell candidate (i.e., a front-end and a back-end candidate are assigned the same (top) score by diagnosis). Thus, there is some ambiguity of whether a standard cell is failing or not in these failing chips. The fifth column ("Total") specifies the size of the dataset analyzed.

Table 5.1 reveals that the frequency of a failing chip being affected with a front-end defect is high. For example, for the dataset associated with the third row, up to 85.6% of the failing

Chip type	Process node (nm)	No. of failing chips: Strong indication of a cell defect (%)	No. of failing chips: Weak indication of a cell defect (%)	Total
Test chip	14	8.8	70.4	11,727
High-volume chip	55	10.2	63.0	1,201
Test chip	28	17.9	85.6	167
Test chip	14	40.1	50.0	1,375

Table 5.1: Summary of silicon test data from chips manufactured across various process nodes and organizations highlighting the percentage of failing chips affected by a front-end defect.

chips may be failing due to an intra-cell defect. For the dataset corresponding to the fourth row, diagnosis suggests that there is a strong indication of a cell-internal defect for 40.1% of the failing chips. Increasing number of front-end defects in advanced process nodes thus warrants the need of a front-end diagnosis technique.

2. The use of large standard cells

Figure 5.1 shows the *bounding circle diameter* (as a measure of physical area) of each cell in a commercial 14nm standard-cell library[1]. The x-axis represents the standard cells that are numbered from 0 to 2245. The y-axis shows the *bounding circle diameter* for each cell. The metric, *bounding circle diameter* is chosen because it directly influences the cost of PFA (Section 4.3.1).

Each standard cell is categorized as either "Primitive", "Complex" (i.e., a custom-designed cell that is logically constructed from a combination of more than one primitive cell, but is physically smaller than the combination due to its efficient transistor-level implementation), and "Sequential" (i.e., a cell that implements sequential logic). Figure 5.1 is divided into three groups, where each group corresponds to a standard cell category.

It is observed from Figure 5.1 that 51.5% of the cells in the library implement custom

[1]Physical-only cells such as filler and decap cells are excluded.

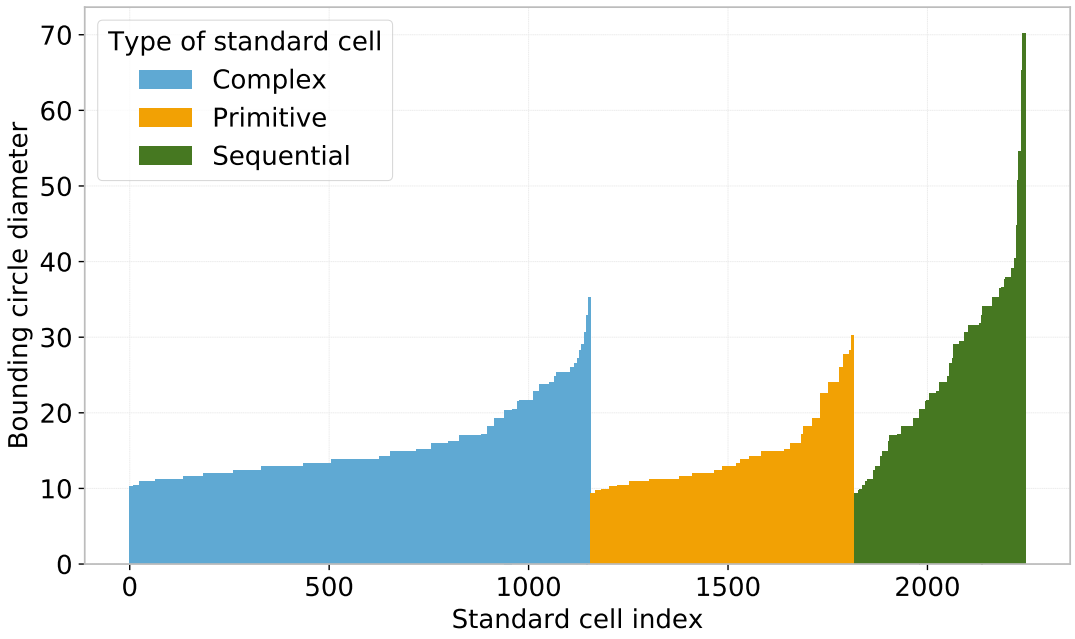


Figure 5.1: Physical area (quantified by *bounding circle diameter* and normalized by the metal-1 pitch) of each cell in a 14nm commercial standard-cell library.

logic, while 19.1% of the cells are sequential. The maximum *bounding circle diameter* for a custom-designed cell is 35.3, primitive cell is 30.2, and sequential cell is 70.1. It should be recalled from Figure 4.31 (in Section 4.3.3) that the minimum *bounding circle diameter* for a failing chip affected with a back-end defect is 1.2, which is 58X smaller than the biggest cell in this library and 8X smaller than the smallest library cell. In addition, *bounding circle diameter* for 73.1% (24.8%) of the back-end candidates is smaller than the biggest (smallest) cell in the library.

Chapter 4 investigates how a back-end layout analysis methodology like PADLOC can successfully enhance physical resolution down to a small portion ("segment") of an interconnect, which, in turn, could decrease the layout area to be searched during PFA. As a consequence, the time and the cost to perform PFA could decrease, which could lead to faster and efficient yield learning.

Thus, if a diagnosis approach is not able to localize a defect within a standard cell, it will increase the cost of PFA significantly. Increasing use of big standard cells in a design with advancing technology thus warrants the need for a diagnosis technique that specializes in locating a defect within a standard cell, which, in turn, reduces the area to be searched during PFA. Furthermore, PFA can become more efficient and cost-effective if a defect inside a standard cell can be identified. This is because a failing chip affected by a front-end defect can be de-layered down to the process layer where the defect is found to likely reside, and all the higher layers could be ignored.

3. Statistical analysis to deduce the underlying root cause

Defect types including poly-contact shorts, poly-contact opens, poly-active shorts, and fin-related defects such as gate oxide shorts can only exist within a standard cell. Those defects would not be identified using a back-end diagnosis technique. Thus, a diagnosis technique is needed that is adept in characterizing a defect inside a standard cell.

Additionally, volume diagnosis statistically correlates diagnoses of a number of failing chips to deduce yield-limiting design/manufacturing issues. A traditional yield-learning method (that is based on the outcome of a back-end diagnosis technique) will fail to identify a systematic failure mechanism affecting a standard cell and could misdirect PFA to inspect an incorrect location in a failing chip. As a result, a considerable amount of PFA resources may be wasted, which likely slows down yield learning.

Therefore, the development of an effective front-end diagnosis technique is imperative to accelerate yield analysis and learning. The next section, Section 5.2, reviews prior work related to the identification of a front-end defect. It also discusses why a generalized approach, instead of a fault model based approach, to defect diagnosis is required, and, subsequently, lays the foundation for NOIDA.

5.2 Prior Work

Many approaches have been put forward over the years to diagnose front-end defects. Work in [163–165] assumes a fault model for defects within the transistor-level description of a cell. Three types of defects are considered in that work – transistor stuck-open defects, transistor resistive-open defects, and bridge defects between each pair of nets in a cell. For each defect type considered in that work, the schematic of a cell is altered by replacing each transistor with a logic sub-circuit. The mapping from a transistor to an equivalent logic sub-circuit is achieved using complex transformation rules and is different for each defect type. A transistor-level cell schematic is thus transformed into a logic netlist (which is separate for each defect type) such that an intra-cell defect in the original transistor-level netlist has an equivalent logic-level fault in the modified logic netlist. Transformation is applied to cells that adhere to the following criteria: the stuck-at fault simulation response at the output of a cell should match the observed circuit response. A logic-level diagnosis tool can then be used on the modified netlist to find the defective cells, and in turn the intra-cell defects.

One main drawback of using the approach presented in [163–165] is that the diagnostic accuracy largely depends on the fault models and the transformation rules. Another disadvantage is that a different model is required for each defect type, which means unknown defect types may go undiagnosed. In addition, that technique is limited to only three defect types; defects such as transistor stuck-closed and net (resistive) open defects are not targeted. Furthermore, precise physical localization of a cell-internal defect is unlikely because cell layout information is not utilized during diagnosis.

Other works that do not locate a defect using a physical netlist of a cell include [166, 167], where a failing transistor is reported using the transistor stuck-open fault model; and [168, 169], where critical path tracing [111–113] is utilized at the transistor level to identify candidate defect locations in a pre-layout schematic netlist of the candidate cell.

An approach is proposed in [170] that focuses on diagnosing an intra-cell bridge defect. Logic-level diagnosis is first used to identify a list of candidates. Each fan-in and fan-out cell corresponding to each candidate location is deemed a candidate cell. Possible locations for a bridge defect in each candidate cell are then identified from a defect extractor tool [79, 278, 365–367]. Next, a pool of defective circuits is created by injecting and simulating a bridge defect at each extracted location. Each defective circuit is then transformed into its equivalent logic sub-circuit [368]. Finally, logic-level diagnosis is utilized to find the logic sub-circuit that explains the tester response, and consequently, the corresponding intra-cell bridge defect. That technique uses a similar idea that is described in [163–165] and hence faces similar drawbacks; its effectiveness mainly depends on how accurately a front-end defect is modeled with a logic fault. It is also limited to the behaviors corresponding to a strong bridge defect. Furthermore, its capability is influenced by the veracity of defect extraction and circuit-level simulation.

In [132, 142–144, 171], possible defective cell locations are identified based on the realistic assumption that the excitation of a cell-internal defect is highly correlated to the input logic values applied to a cell. Logic values at the inputs of each candidate cell are collected for Tester-Fail-Simulation-Fail (TFSF) patterns, that is, patterns that fail on the tester and propagate the error effects from the defect location to the circuit outputs; and Tester-Pass-Simulation-Fail (TPSF) patterns, that is, patterns that pass on the tester but propagate the error effects from the defect location to the circuit outputs. Such input-value combinations will henceforth be referred to as cell-level failing and passing patterns. Cells with inconsistent input conditions, i.e., input conditions that appear in both cell-level passing and failing patterns, are discarded from further analysis. This form of consistency checking is a special case of the approach described in Section 4.1.2. For the remaining consistent cells, the input conditions are matched with a fault dictionary. The fault dictionary is created by performing a switch-level simulation of various intra-cell defects. However, only transistor defects and

physically-feasible bridge defects are considered in that work; defects such as net open defects (resistive or otherwise) are not targeted. In addition, the use of a fault dictionary implies that an unknown defect would remain undiagnosed.

The methodology described in [172–174] follows a similar approach as [171]. It employs consistency check to find an initial set of candidate cells and then compares the cell input conditions with a fault dictionary to identify the fault that explains the observed circuit response. In contrast to the work of [171], it performs transistor-level simulation (with extracted parasitics) for each potential defect to build the dictionary [369]. Realistic defect locations are extracted from the layout as well. However, it restricts itself to diagnosing an intra-cell bridge, a transistor stuck-open and an intra-cell (resistive) contact open defect. Moreover, similar to the work of [170], the quality of diagnosis greatly depends on possibly inaccurate defect extraction and transistor-level simulation.

Work presented in [73–78] builds a fault dictionary as well using transistor-level simulation. Relative to other fault model based approaches discussed up to this point, it expands the front-end defect universe considered to create a fault model. It models an open defect at any netlist component (i.e., a net, via or a contact), a bridge defect between two adjacent components and a transistor defect, with varying strengths. The fault model has been used for cell-internal diagnosis in [175–179]. However, similar to the works of [170, 172–174], the efficacy of the approach used in [175–179] relies on the accuracy of the SPICE models and fortuitous identification of an *unseen* defect.

Authors in [180] propose a DFM-aware fault model where probable defect locations within a cell are identified using DFM guideline violations (instead of using extracted parasitics). A DFM guideline represents a vulnerability in the manufacturing process and its violation is thus more likely to cause a defect. In [79,80,366,367], on the other hand, possible defects are identified at the process-level by taking into account prior statistical information on defect likelihood that is obtained from the manufacturing process. Each defect is then mapped to a

faulty behavior at the circuit-level. The method outlined in [181] goes one abstraction level below by modeling a defect at the contamination[2] level itself. However, these techniques face similar drawbacks to any other model-based approach.

Methods reviewed up to this point either devise a new fault model or employ an existing model to diagnose a front-end defect. A model-based approach stores pre-computed defect behaviors in a fault dictionary. While a fault dictionary is impractical to use during back-end diagnosis due to its space and time complexities, it is tractable during front-end diagnosis because the fault simulation responses are generated and stored for a library of standard cells (and not for the entire design, which is much larger in size) once per technology (and not per design). In addition, fault models enable defect behavior characterization. However, as technology progresses, new materials and fabrication steps are employed, which results in new defect mechanisms. New defect mechanisms (e.g., fin-related defects in FinFET-based circuits [370]) can create misbehaviors that are not sufficiently captured by existing fault models [371]. Thus, the unpredictability of defects necessitates the need for a more generalized approach to defect diagnosis.

Even if fault models accurately captured the typical behaviors of the targeted defects, circuit-level noise can cause deviations between the predicted behavior and the behavior observed on the tester. Various sources of noise exist in digital circuits that include process variation, crosstalk signal noise, power supply noise, and substrate-coupling noise [372]. Deviations between the observed and the predicted behavior can also result from inaccurate SPICE models used for defect extraction and transistor-level simulation employed for formulating fault models. Depending on the amount of deviation, a weak logic value at a cell output, due to an intra-cell defect, can be interpreted as a logic-1 or a logic-0 by a receiver cell (depending on its switching threshold, which too is affected by noise), and can result in a

[2]A contamination is defined as an accidental particle that is deposited on a chip during manufacturing, which may or may not induce a defect.

tester response that is not predicted by the corresponding fault. Thus, circuit-level noise and the resulting ambiguity in signal logic values warrant the need for a noise-resistant front-end diagnosis approach.

5.3 Diagnosis Methodology

The discussion of prior work associated with the diagnosis of a defect within a standard cell using various layout and circuit parameters in Section 5.2 suggests that an ideal front-end diagnosis approach should possess the following characteristics:

1. Physical localization of a defect within a standard cell (in terms of its $x-y-z$ location).
2. Defect modeling with minimal assumptions.
3. Characterization of a defect with respect to its behavior.
4. Robust to circuit-level noise

NOIDA attempts to fulfill the aforementioned objectives. NOIDA circumvents the problem of potentially inaccurate defect modeling observed in prior work by avoiding the use of a fault dictionary. Instead, it derives the defect behavior by analyzing the logic activity of the intra-cell nodes surrounding the likely defect location. Here, a defect is assumed to be localized and controlled by the circuitry in close proximity. This assumption holds true for a variety of defects including intra-cell bridge, open and transistor defects. The nodes near a candidate are collectively referred to as its neighborhood; the logic values applied to the nodes in the neighborhood form a neighborhood state. Changes in neighborhood state over time can be recorded for sequence- and timing-dependent defects. The output of NOIDA is a set of intra-cell candidates, where each candidate is a tuple consisting of its physical location and its likely behavior (and consequently, its defect type).

Figure 5.2 shows an overview of NOIDA. The input to NOIDA is a set of candidate cells that have been deduced by a logic diagnosis methodology such as LearnX/MD-LearnX. The flow begins by deriving all the possible locations for an intra-cell defect within each candidate cell. Cell-level passing and failing patterns are then identified by performing a transistor-level simulation of "stuck-at" faults at each candidate node. The next step, called *intra-cell node neighborhood identification*, identifies the physical neighborhood for each candidate. A *consistency check* is then performed on each candidate node to identify nodes that portray consistent pass-fail behavior. Consistent candidate nodes are then mapped to one or more defect candidates, depending on the custom fault model extracted by NOIDA. Defect candidates can then be ranked using critical area, DFM violations, etc.

5.3.1 Intra-cell Node Identification

A physical transistor-level description of a cell (such as its physical layout and SPICE netlist with extracted parasitics) is used to identify intra-cell nodes. There are different ways to identify the nodes in a cell. In a SPICE netlist, for example, each location where more than one SPICE component (such as transistors and extracted parasitics) is connected is deemed a node.

The schematic of an inverter is used to illustrate node identification in Figure 5.3. The circuit consists of six resisters, $R_1 - R_6$, ten capacitances $C_1 - C_{10}$ and two transistors $M_1 - M_2$. There are ten internal nodes identified for this cell, namely, $\{A, A_1, A_2, Z, M_1 : g, M_1 : d, M_1 : s, M_2 : g, M_2 : d, M_2 : s\}$.

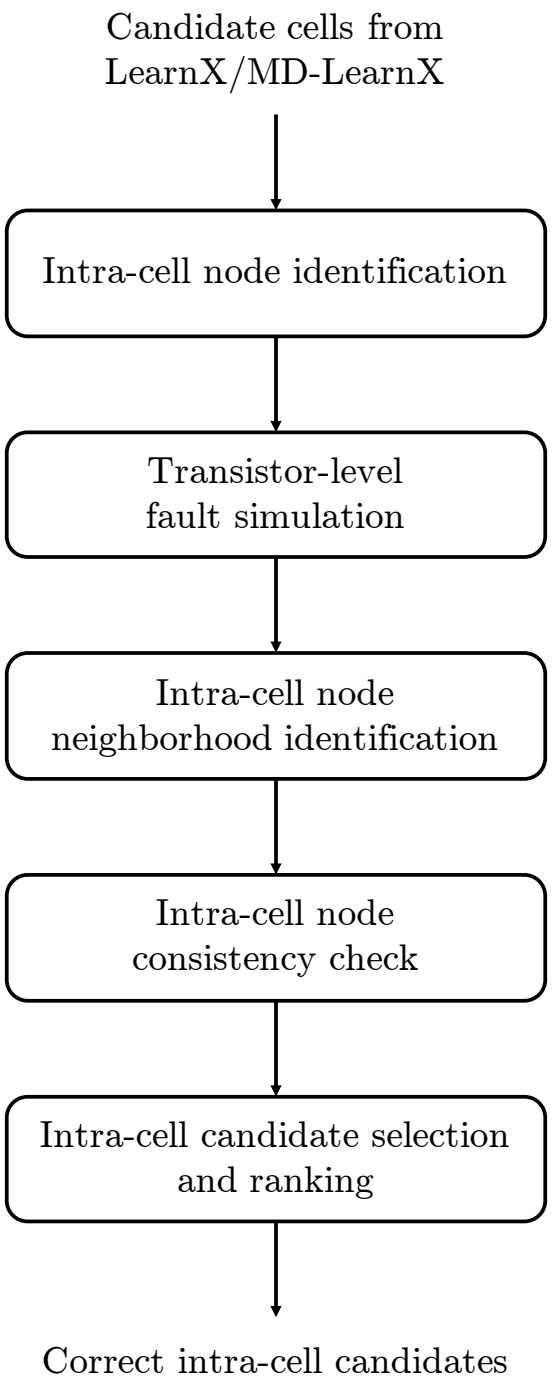


Figure 5.2: An overview of the proposed front-end diagnosis methodology, NOIDA.

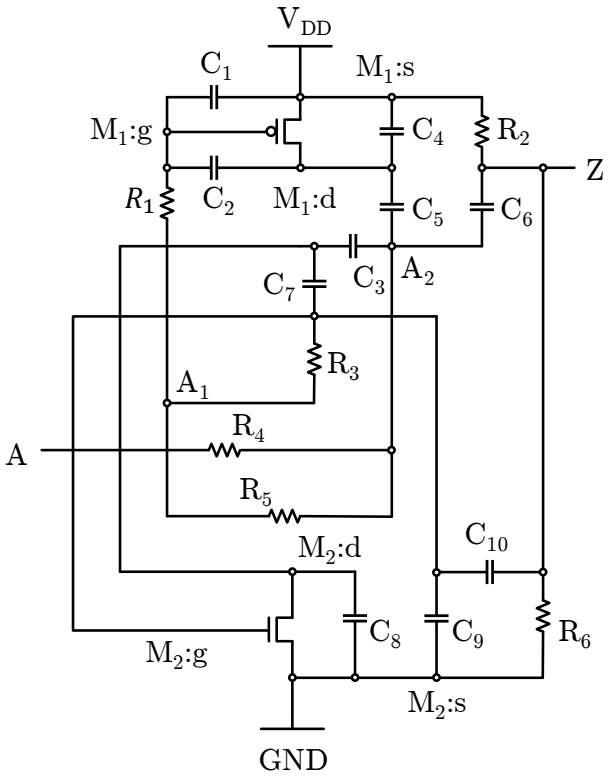


Figure 5.3: A schematic view of an inverter cell with parasitics extracted. Parasitics affecting power rails are not shown for clarity.

5.3.2 Transistor-level Fault Simulation

Each intra-cell node is faulted at the opposite value of the expected value for each cell-level failing and passing pattern[3]. This is achieved by adding a near-zero resistor between each node and V_{DD} (or GND, depending on the fault value) in the SPICE netlist. The altered SPICE netlist is then simulated with an analog simulator for the cell-level failing and passing patterns. Each simulation response is digitized using the following criteria: the logic value at a cell output is deemed a logic-1 (logic-0) if the output voltage is more (less) than half of the supply voltage. This logic threshold value depends on the process technology and can be specified by the user. The cell-level passing (failing) patterns for which a faulted node

[3]The cell-level failing and passing patterns, which have been defined in Section 5.2, can be obtained via any logic-level diagnosis technique.

produces and propagates an error to the cell output are the cell-level TPSF (TFSF) patterns for the faulted node. Each faulted node with at least one cell-level TFSF pattern is adjudged an initial diagnosis candidate.

Figure 5.4 illustrates fault simulation of the intra-cell node, $M_2 : g$ (shown with a green oval around it in the figure), for the input pattern $A = 0$. The analog voltage obtained at each intra-cell node from fault simulation is shown in a dotted yellow box. The expected voltage at each node for the applied pattern is shown in a dashed blue box. Because the error is propagated to a cell output (the simulated and the expected cell output voltages are 0V and 1V, respectively), the applied pattern would be deemed a cell-level TPSF (TFSF) pattern if it is a cell level passing (failing) pattern.

5.3.3 Intra-cell Node Neighborhood Identification

The core assumption in NOIDA is that the activation of a cell-internal defect is influenced by the nodes in its neighborhood. The behavior of a defect can then be determined by identifying the neighborhood that controls defect activation. If the SPICE netlist with extracted parasitics is used, then the neighborhood of a node constitutes all nodes that are coupled to it by capacitors. This is a reasonable way to identify neighbors and adheres to the localization assumption. For the inverter illustrated in Figure 5.3, the neighborhood for each node is shown in Table 5.2.

Neighborhood of a node can also be computed by using Design Rule Check (DRC) rules, Design-for-Manufacturability (DFM) guidelines and geometric proximity analysis [125, 142–144, 158, 161, 278, 280].

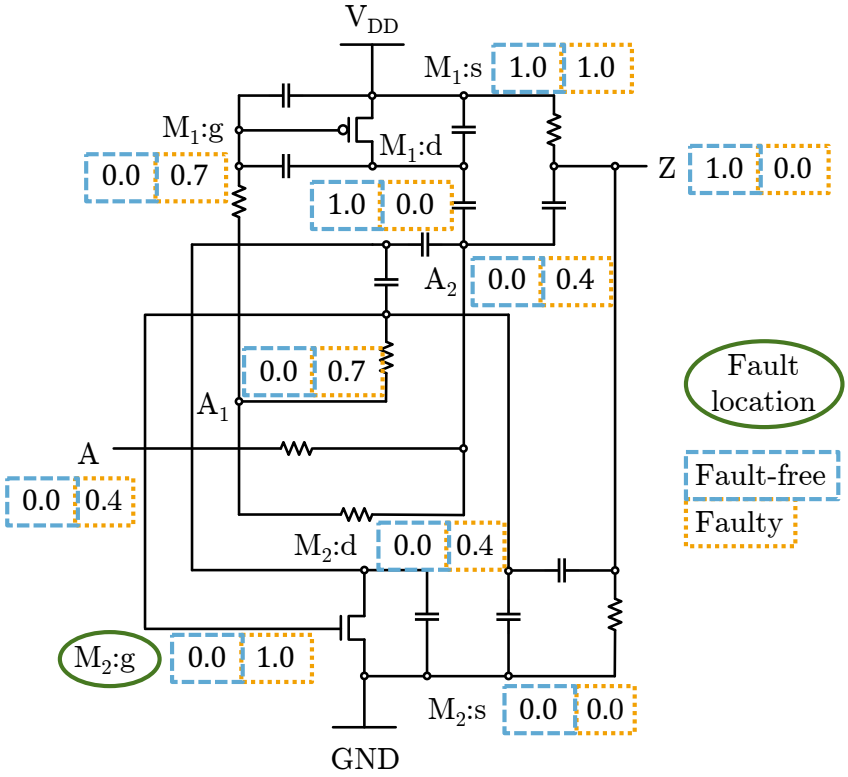


Figure 5.4: Analog voltage obtained at each intra-cell node when an inverter is simulated by adding a near-zero resistor between $M_2 : g$ and V_{DD} (to emulate $M_2 : g$ stuck-at 1) for $A = 0$.

5.3.4 Intra-cell Node Consistency Check

Analog voltages at each cell-internal node are stored during fault simulation (Section 5.3.2). Each value is then converted into its logic equivalent by using the same logic threshold value mentioned in Section 5.3.2. Once the analog value at each node is digitized, the procedure of determining the neighborhood state for a node for a pattern is similar to DIAGNOSIX and PADLOC (Section 4.1.2 and Section 4.2.2, respectively). For static defects, the neighborhood state for a candidate is the set of logical values established on the neighborhood nodes for the pattern applied. For sequence-dependent defects, the neighborhood state tracks the logical values for two or more patterns.

Intra-cell node	Intra-cell neighborhood
A	–
A_1	–
A_2	$M_1 : d$, $M_2 : d$, Z
$M_1 : d$	$M_1 : g$, $M_1 : s$, A_2
$M_1 : g$	$M_1 : d$, $M_2 : s$
$M_1 : s$	$M_1 : d$, $M_2 : g$
$M_2 : d$	$M_2 : g$, $M_2 : s$, A_2
$M_2 : g$	$M_2 : d$, $M_2 : s$, Z
$M_2 : s$	$M_1 : d$, $M_2 : g$
Z	$M_2 : g$, A_2

Table 5.2: Intra-cell neighborhood for each intra-cell node in the inverter that is illustrated in Figure 5.3. Each neighbor of a node is capacitively coupled to it.

For instance, the neighborhood state for node $M_2 : g$ for the pattern $A = 0$ is 000, which is calculated after digitizing the values that are obtained from the transistor-level simulation illustrated in Figure 5.4.

The neighborhood state for each intra-cell candidate is analyzed for cell-level TPSF and TFSF patterns. If the neighborhood state for a candidate is the same for any pair of cell-level TPSF and TFSF patterns, then the candidate has demonstrated inconsistency.

For instance, if the failing state (i.e., the neighborhood state established for a cell-level TFSF pattern) for node $M_2 : g$ in the inverter schematic shown in Figure 5.3 is 000 and its passing state (i.e., the neighborhood state established for a cell-level TPSF pattern) is 001, then $M_2 : g$ is deemed consistent. However, if the passing state is 000 as well, then it is adjudged inconsistent.

5.3.5 Intra-cell Candidate Selection and Ranking

Each candidate that is found to be consistent is further examined. Because the underlying assumption in NOIDA is that the neighborhood governs the value at a defect location, a

customized fault model can be extracted for each candidate that characterizes its behavior. A candidate node (or a group of nodes, if the suggested behavior resembles that of a bridge or a transistor defect, for example) is then mapped to a defect candidate. Thus, in contrast to prior work, NOIDA infers the behavior of a defect and consequently its type from its neighborhood.

Next, a minimal group of defect candidates are selected that can jointly explain the cell output(s) for each failing pattern. Each candidate cover can then be ranked based on the amount of consistency of its constituent candidates and the number of unexplained passing patterns (i.e., cell-level TPSF patterns). A defect candidate in a cover can also be graded on the basis of its likelihood of representing a real defect. For example, a weighted critical area based approach is suggested in [80] to measure the probability of a defect. DFM guidelines can also be utilized to identify likely intra-cell defect candidates [180].

To summarize, NOIDA assumes that the behavior of a defect is influenced by the circuitry within some distance d surrounding it. It derives the defect behavior by analyzing the nodes surrounding its location, instead of being limited to a pre-computed fault dictionary. In contrast to prior work that essentially employs a lookup table to search for a fault that matches the observed behavior, NOIDA utilizes an "effect-cause" approach where the cell response is examined to find a group of candidates that collectively explains the observed behavior. As a consequence, NOIDA is competent to localize a defect whose behavior cannot be predicted by an existing front-end fault model. More importantly, no assumptions are made regarding the number of intra-cell nodes affected by a defect, or the number of defects.

The performance of NOIDA primarily depends on the effectiveness of two steps, namely, intra-cell node neighborhood identification and consistency check. The possibility of how a correct candidate can be eliminated by either of these steps is as follows.

1. **Neighborhood identification**:As described in Section 5.3.3, there are different methods to identify the physical neighborhood of a node. For example, if the neighborhood

of a node is assumed to consist of all nodes that are capacitively coupled to it, then the effectiveness of NOIDA depends on the accuracy of the potentially inexact extracted capacitances. If geometric proximity is used to find the neighborhood of a node, then it is possible that, depending on the size of the neighborhood, irrelevant neighbors (if the neighborhood size is too large) or only a subset of neighborhood (if the size is too small) is identified.

2. **Amount of (in)consistency**: The amount of (in)consistency can be modulated for candidate elimination. It should be recalled that a node is eliminated from further analysis based on the cardinality of the intersection set of passing and failing neighborhood states. As pointed out in Section 4.3.2.2, an aggressive approach would eliminate a candidate if a failing state is identical to any passing state. A conservative approach, on the other hand, would discard a candidate if every failing state matches a passing state. It considers the feasibility of a failing state being incorrect due to factors such as an inaccuracy arising from estimating voltages at each node from a SPICE simulation and imprecise identification of the neighborhood of a node. Naturally, a conservative approach yields more candidates and hurts physical resolution, while increasing the likelihood of retaining the correct candidate. The trade-off among the amount of consistency, physical accuracy and physical resolution should be carefully analyzed for optimum effectiveness.

Improving the performance of NOIDA by focusing on these two aspects remains as future work and is further discussed in Chapter 6.

5.4 Experiments

This section describes several experiments that are conducted to validate NOIDA and compare its potential with commercial diagnosis. NOIDA is evaluated on a FinFET-based 7nm

standard-cell library [373].

The rest of this section is organized as follows. Section 5.4.1 discusses the diagnostic metrics employed to measure the quality of diagnosis achieved by NOIDA. Section 5.4.2 describes the setup for the experiments conducted. Section 5.4.3 presents the experiment results when an exhaustive test set is used for diagnosis. The consequences of using a test set generated by a commercial intra-cell ATPG tool are investigated in Section 5.4.4. Section 5.4.5 studies the effect of introducing noise to the tester response on the quality of diagnosis.

5.4.1 Diagnostic Metrics

The output of NOIDA is a set of consistent intra-cell defect candidates, where each candidate is a tuple consisting of its physical location ($x - y - z$ location) and its likely behavior. The output of commercial diagnosis is regarded here as a set of highest-ranked candidates among the reported candidates.

As discussed in Section 1.2, a diagnosis methodology should ideally report a single correct candidate that precisely represents and captures the behavior of the defect. Thus, the following metrics are employed to evaluate the effectiveness of NOIDA. The output of NOIDA is also compared with state-of-the-art commercial diagnosis software to discover its capability.

- Physical accuracy: It is a binary measurement of whether the location and the type of an intra-cell defect is correctly pinpointed by diagnosis. Specifically, a transistor defect is deemed accurate if diagnosis reports the defective transistor; an open defect is judged accurate if one of the intra-cell nodes associated with the defect is correctly suspected; and a bridge defect is considered accurate if both the bridged intra-cell nodes are correctly identified.

- Physical resolution: It is the number of defect candidates returned by diagnosis.

- *Home run*: A diagnosis approach is said to hit a home run when a single defect candidate is reported correctly.

5.4.2 Setup

To validate NOIDA, each cell in the 7nm standard-cell library is individually analyzed. A pool of defective cells is created by injecting cell defects (one at a time) into the layout of each cell. The defects injected include opens, bridges (feedback and non-feedback), stuck-open and stuck-closed transistors with varying strengths. The resistance values range from 1Ω to $20k\Omega$ for bridge and transistor stuck-closed defects, and from $1G\Omega$ to $1k\Omega$ for open and transistor stuck-open defects. For each defective cell layout, a corresponding transistor-level netlist is extracted. Because the behavior of a defect is unknown, i.e., whether it is static or sequence-dependent, analog simulation is performed on each altered netlist using an exhaustive two-pattern test set. A defect is considered detected if the voltage at a cell output deviates from its expected, defect-free value of V_{DD} or GND by more than 50%. Also, and importantly, patterns that detect the defect are deemed cell-level failing patterns, while the remaining patterns are treated as cell-level passing patterns.

Each defect simulation response is analyzed to determine if it exhibits static or sequence-dependent behavior. A sequence-dependent defect requires a sequence of patterns (two patterns in this case) for detection. Detection of a static defect is independent of the first pattern applied for all two-pattern combinations. In other words, a static defect is detected by a single pattern.

Figure 5.5 shows the number of defects that are analyzed for each primitive standard cell in the library, while Figure 5.6 shows the distribution for each complex cell. Each figure consists of two plots – each showing the distribution of static and sequence-dependent defects as a proportion of the total number of defects examined for a cell. A total of 115 standard cells are inspected, where 48 (41.7%) are primitive and 67 (59.3%) are complex cells. A total

of 34,330 defects are analyzed, out of which 10,303 (30.0%) defects have been examined for a primitive cell while 24,027 (70.0%) for a complex cell.

The following observations can be made from Figures 5.5 and 5.6.

1. Figure 5.5 reveals that out of 10,303 defects, 6,626 (64.3%) defects are identified to the static while 3,677 (35.7%) are found to be sequence-dependent. Figure 5.6 indicates that among 24,027 defects, 10,162 (42.3%) defects are static while 13,865 (57.7%) seem to be sequence-dependent. This means that a defect is likely to be detected by a sequence of patterns if it resides in a complex standard cell; specifically, 79.0% of the analyzed sequence-dependent defects reside in complex cells.

2. It is seen from Figure 5.6 that the cell "FAX1", which implements the functionality of a full adder, contains the highest number of possible defects (751 defects).

3. Among primitive cells, the 4-input NOR gate with a high drive strength, referred to as "NOR4XP75" in Figure 5.5, appears to have the most number of possible defects (338 defects).

Among primitive cells, Figure 5.7 and Figure 5.8 show the distribution of static and sequence-dependent defects, respectively, based on their type (i.e., whether a defect to be examined is an open, a bridge or a transistor defect). Figures 5.7 and 5.8 are divided into three plots each, one corresponding to each defect type. The gray bar in each plot denotes the number of static (sequence-dependent) defects in Figure 5.7 (Figure 5.8). Figure 5.7 reveals that, among static defects, 50.8% are bridge defects, 19.8% are open defects and 29.4% are transistor defects. On the other hand, it is seen from Figure 5.8 that only 12.5% of the sequence-dependent defects are bridge defects, while the number of open and transistor defects are nearly similar (45.9% and 41.7%, respectively). It is also observed that a majority of bridge defects (88.0%) are static.

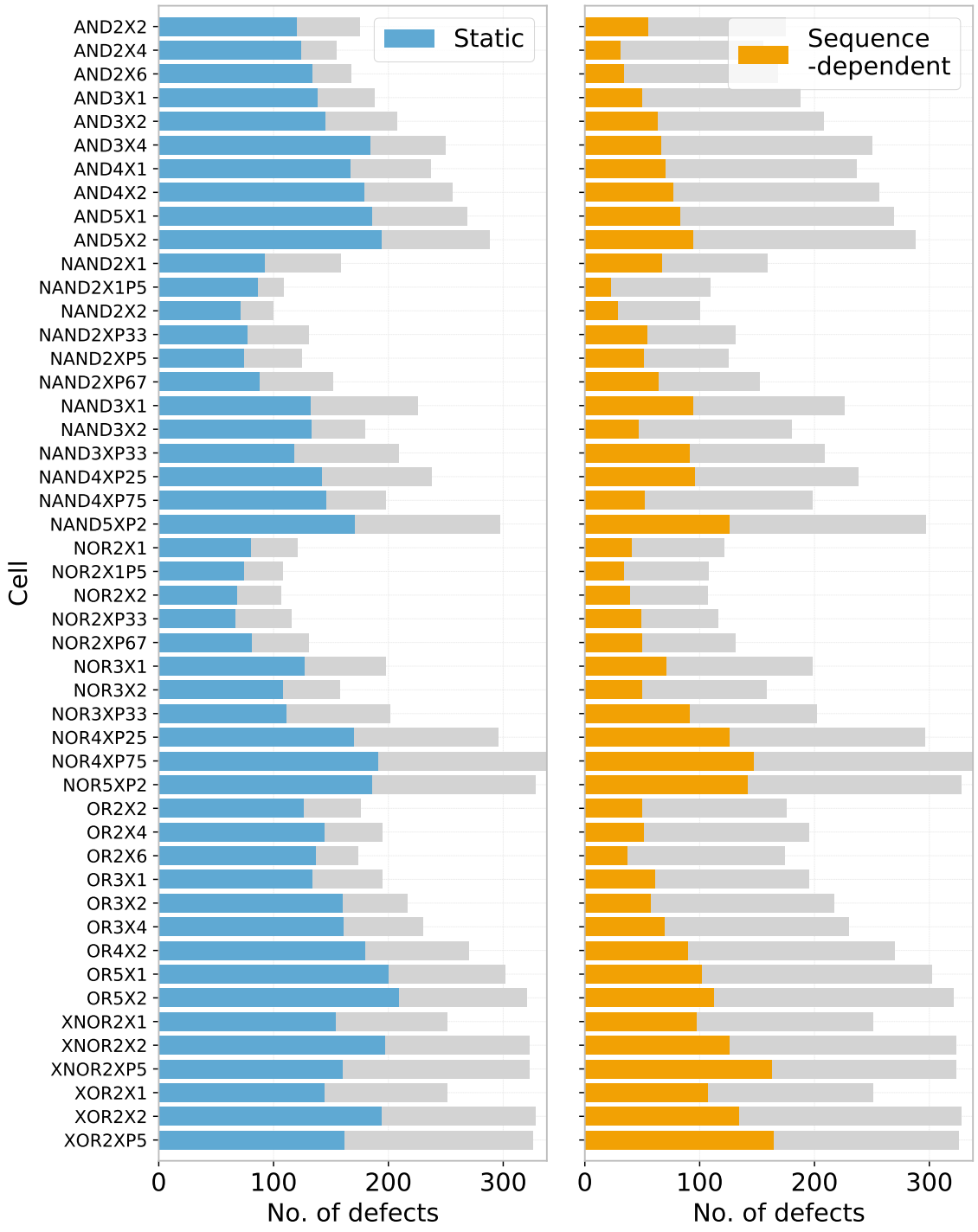


Figure 5.5: Distribution of static and sequence-dependent defects for primitive cells in a 7nm standard-cell library.

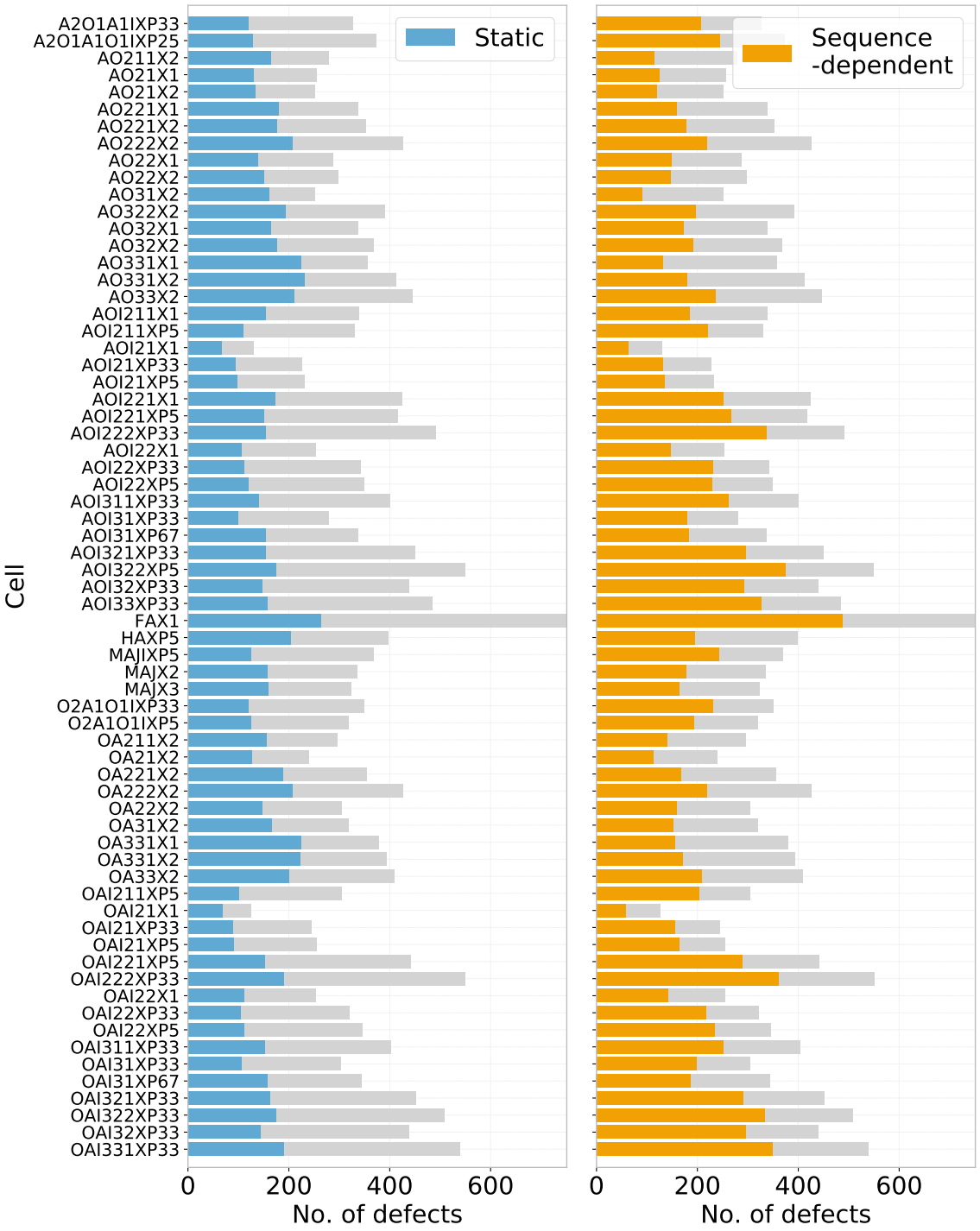


Figure 5.6: Distribution of static and sequence-dependent defects for complex cells in a 7nm standard-cell library.

Similarly, Figure 5.9 and Figure 5.10 show the distribution of static and sequence-dependent defects, respectively, based on their defect type, for complex cells. Figure 5.9 reveals that, among static defects, 65.0% are bridge defects, while only 13.8% are open defects. On the other hand, it is seen from Figure 5.10 that just 19.9% of the sequence-dependent defects are bridge defects, while the number of open and transistor defects are 42.8% and 37.3%, respectively. It is also observed that a majority of bridge defects (70.5%) are static, and most of the open and transistor defects are sequence-dependent (80.8% and 70.6%, respectively).

The experiments presented in the subsequent subsections (Sections 5.4.3 through 5.4.5) evaluate NOIDA and compare its performance with commercial diagnosis. Commercial diagnosis uses a pre-computed fault dictionary to diagnose a front-end defect. In the following experiments, to mimic the ever-evolving nature of defects, it is assumed that the fault model employed by commercial diagnosis does not capture the behavior of every injected defect. Specifically, here, it is assumed that weak resistive defects are not modeled by commercial diagnosis. Thus, a bridge or a transistor stuck-closed defect with resistance between $10k\Omega$ to $20k\Omega$, or an open or a transistor stuck-open defect with resistance between $1k\Omega$ to $10k\Omega$ is not considered while constructing the dictionary employed by the commercial tool.

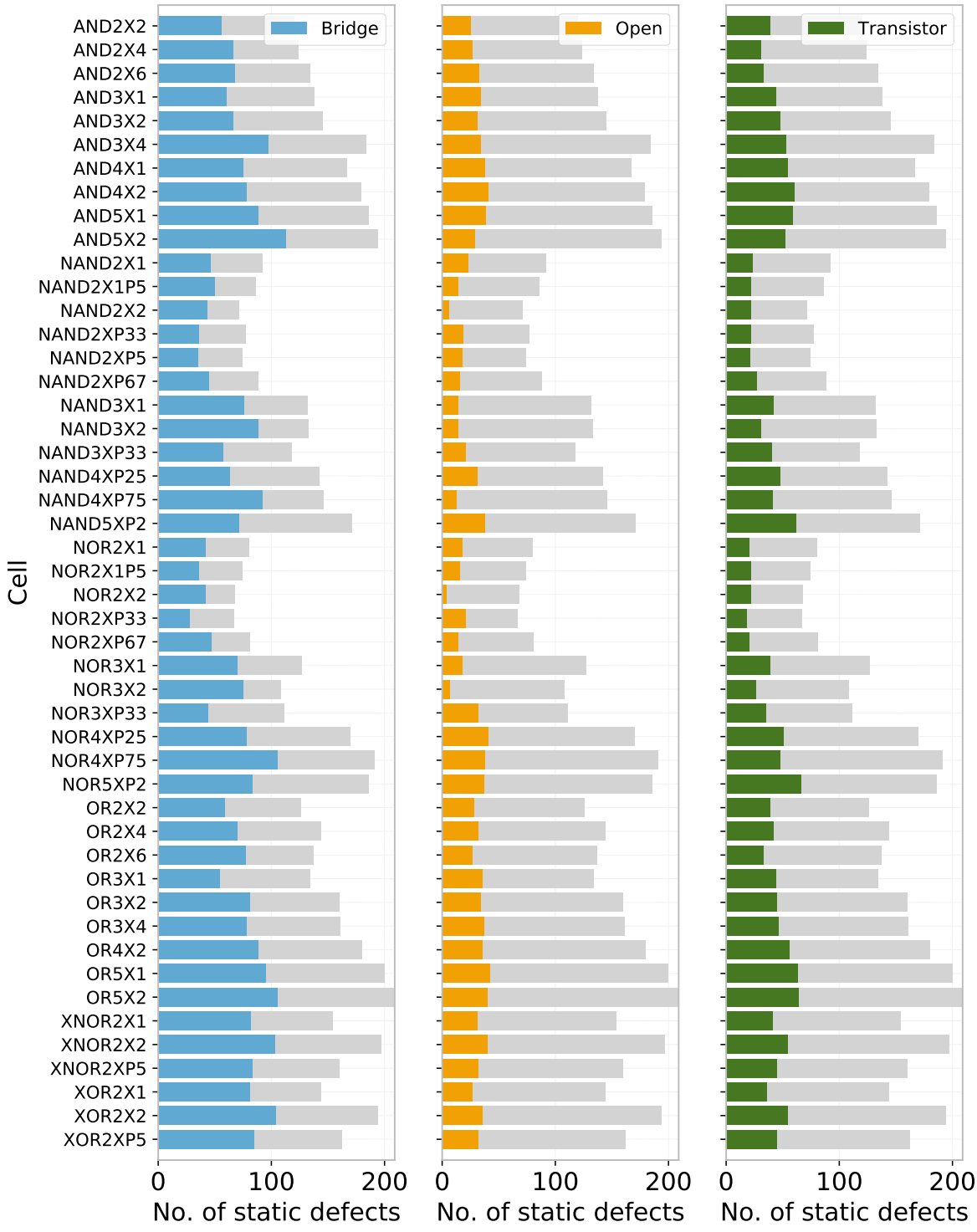


Figure 5.7: Distribution of static defects by defect type for primitive cells in a 7nm standard-cell library.

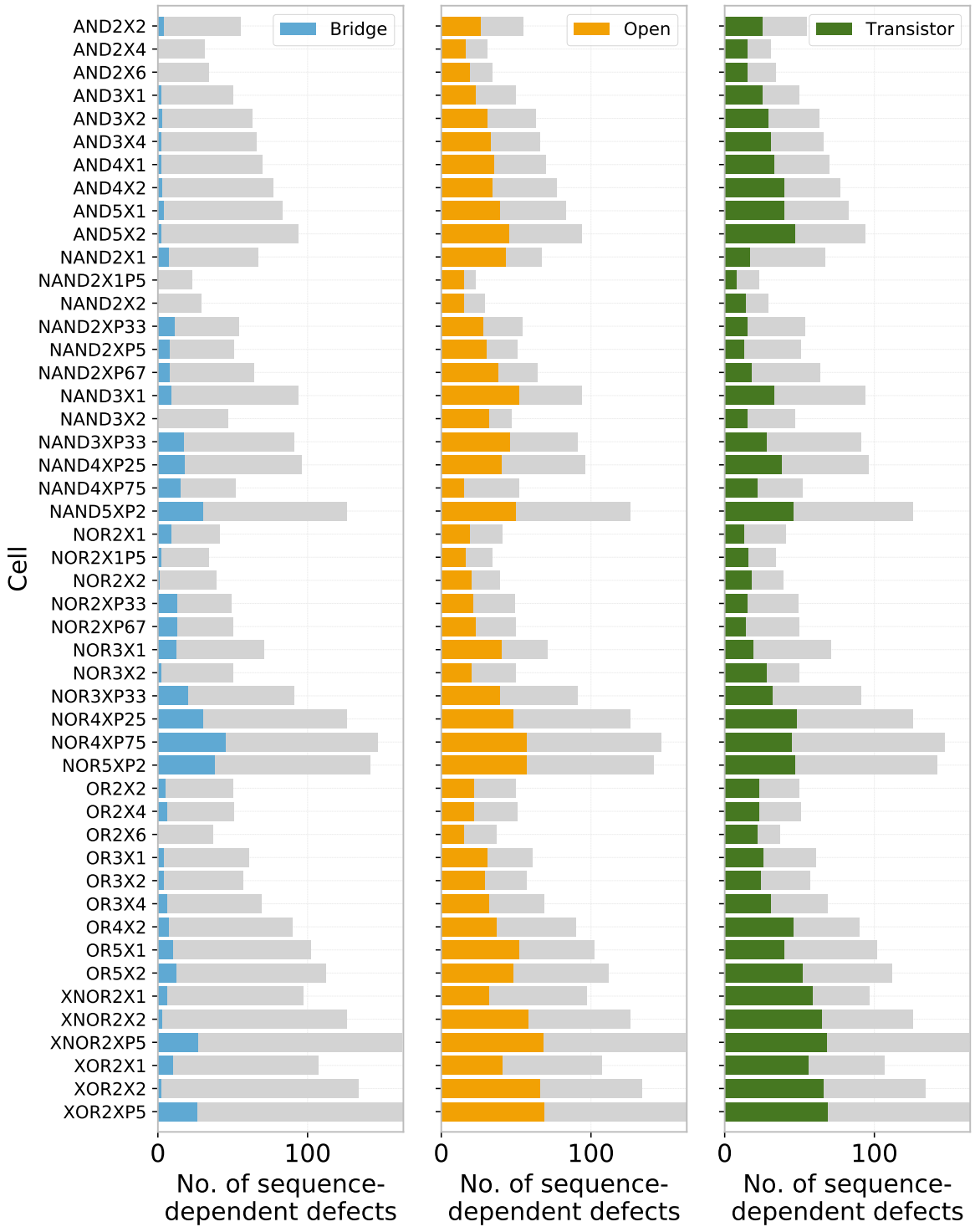


Figure 5.8: Distribution of sequence-dependent defects by defect type for primitive cells in a 7nm standard-cell library.

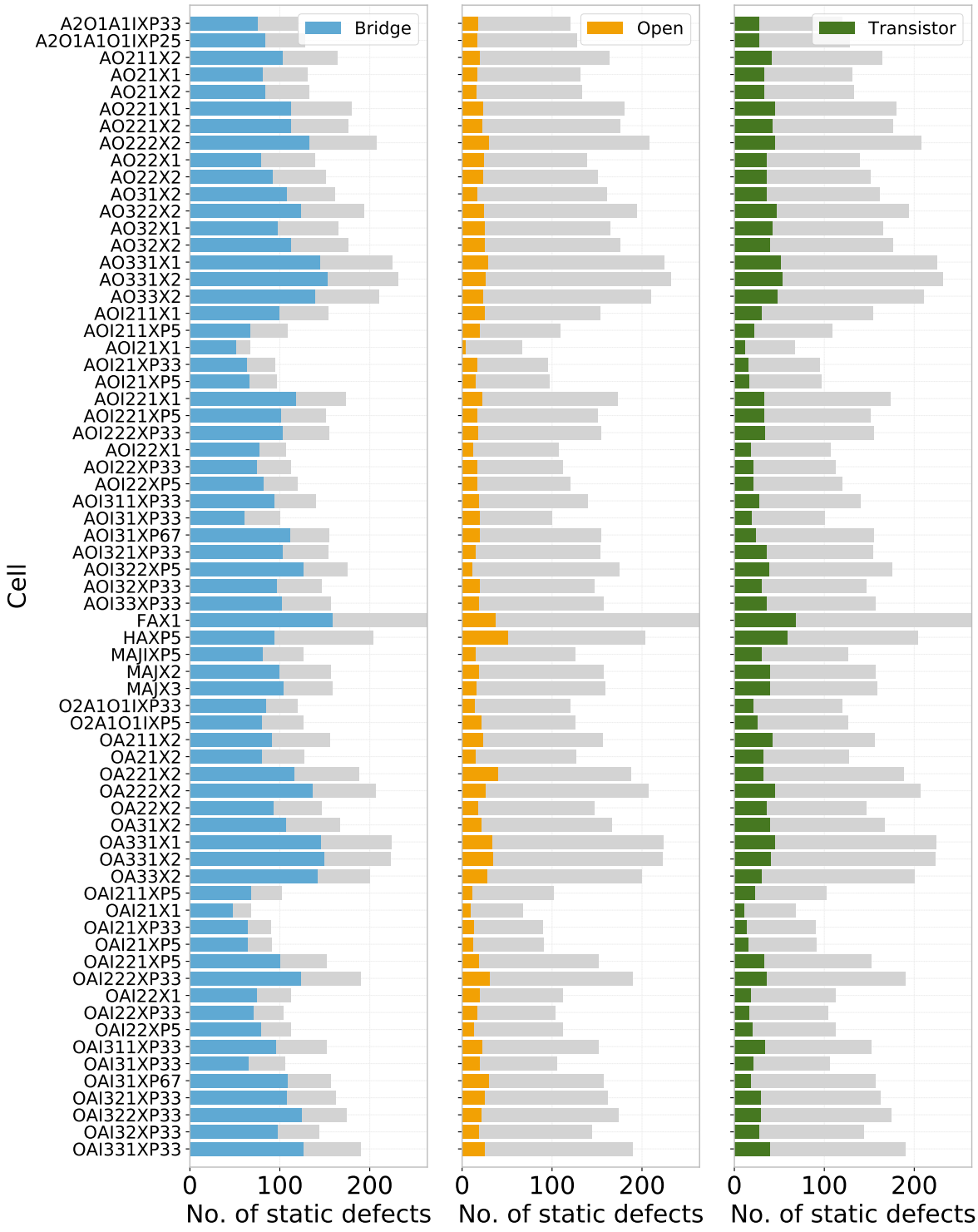


Figure 5.9: Distribution of static defects by defect type for complex cells in a 7nm standard-cell library.

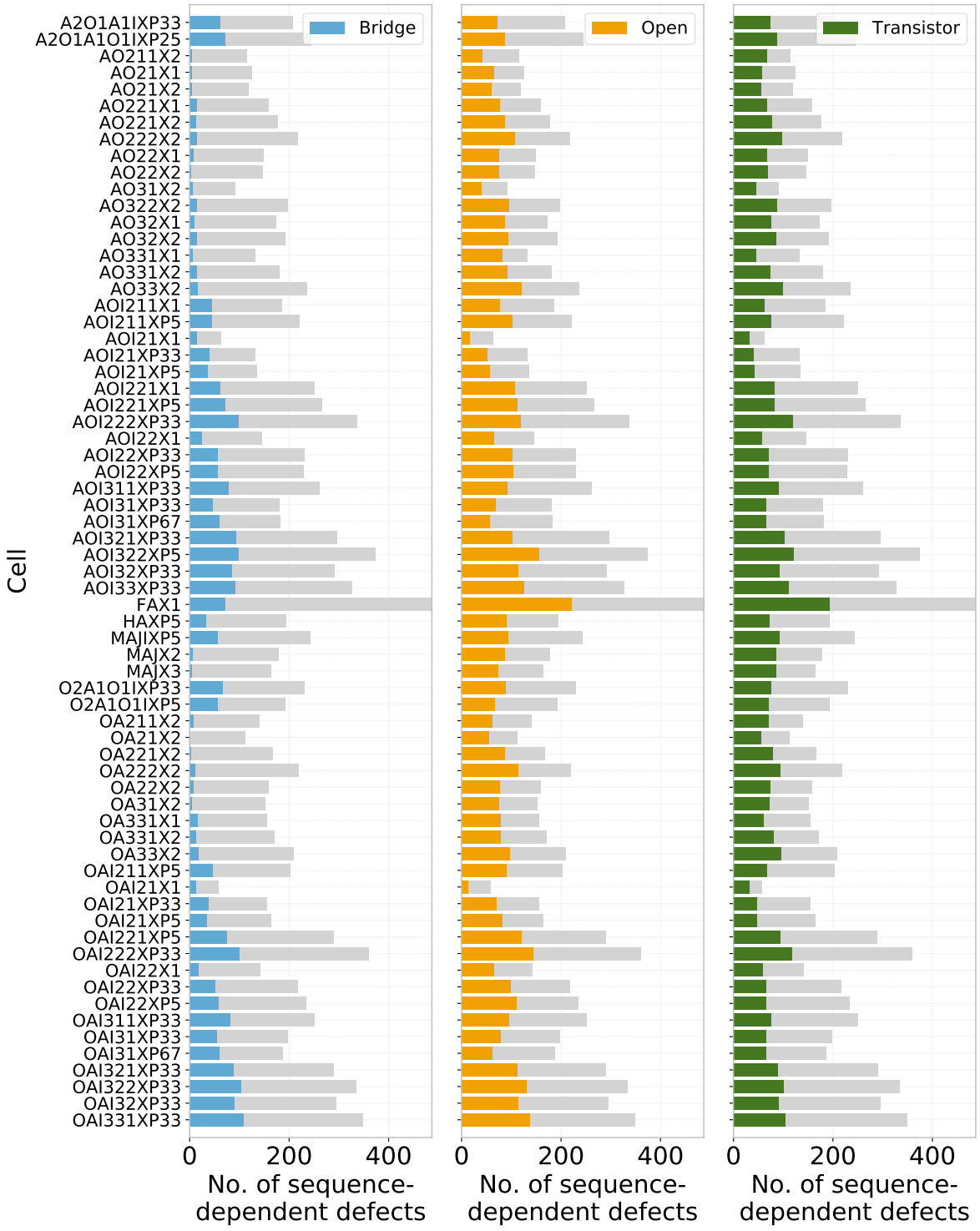


Figure 5.10: Distribution of sequence-dependent defects by defect type for complex cells in a 7nm standard-cell library.

5.4.3 Results: Exhaustive Testing

In this experiment, NOIDA is applied to each defective cell while assuming that each cell is tested exhaustively[4]. NOIDA is evaluated using the diagnostic metrics discussed in Section 5.4.1 and its performance is compared with commercial diagnosis.

Figure 5.11 shows the average physical accuracy for each primitive cell. The figure is partitioned into two plots for clarity. The x-axis shows the percentage of defects diagnosed correctly by a diagnosis approach. In each plot, each horizontal bar is associated with a standard cell (whose name is displayed within its corresponding horizontal bar) and is divided into two parts; the "right" part (i.e., towards the positive x axis) represents the accuracy achieved by NOIDA, while the "left" part (i.e., towards the negative x axis) represents the accuracy attained by commercial diagnosis. Each value written on either side of a horizontal bar is the accuracy achieved by the corresponding standard cell. For each cell, its accuracy is written in "blue" ("yellow") if NOIDA (commercial diagnosis) correctly diagnoses a higher percentage of defects.

Figure 5.11 reveals that NOIDA performs better than state-of-the-art commercial diagnosis for every primitive cell. On average, the physical accuracy for NOIDA is 99.2%, which is 13.5% higher than commercial diagnosis. Out of 48 primitive cells, 33 (68.7%) cells achieve perfect physical accuracy when NOIDA is used. Moreover, the maximum improvement in accuracy is observed for the cell, "NOR2X2"; while NOIDA correctly diagnoses each defective cell of that type, commercial diagnosis correctly diagnoses only 71.0% of defects. Further investigation reveals that NOIDA achieves 99.7% (98.1%) accuracy for static (sequence-dependent) defects, which is an improvement of 15.4% (16.5%) over commercial diagnosis.

Figure 5.12 shows the average physical accuracy for each complex cell. It is clearly seen

[4]The size of an exhaustive test set used for the diagnosis of a static defect within a n-input standard cell is 2^n, and $(2^n)(2^n - 1)$ for a sequence-dependent defect.

from the figure that NOIDA outperforms commercial diagnosis for each complex cell as well. On average, the physical accuracy for NOIDA is 95.4%, which is 15.0% higher than commercial diagnosis. Out of 67 complex cells, 48 (71.6%) cells achieve physical accuracy greater than 95.0%, while the maximum physical accuracy attained by commercial diagnosis is 92.8%. In addition, the maximum improvement in physical accuracy is observed for the cell, "AO31X2", where the accuracy for NOIDA is 94.0% and the accuracy for commercial diagnosis is 65.0%. Further investigation reveals that NOIDA achieves 98.2% (92.3%) accuracy for static (sequence-dependent) defects, which is an improvement of 17.6% (19.7%) over commercial diagnosis.

Figures 5.13 and 5.14 show histograms of the physical resolution for primitive and complex cells, respectively. The x-axis shows the resolution and the y-axis shows the number of defects. The percentage of defects accurately diagnosed for a particular resolution is shown at the top of its corresponding plot-bar. The top half of each figure (i.e., above $y = 0$) shows the distribution of the number of defects that are accurately diagnosed while the bottom half shows the distribution of the inaccurately diagnosed defects. The percentage of defects diagnosed correctly by each diagnosis technique is shown above the plot in each figure.

It is observed from Figure 5.13 that, among the defects diagnosed with perfect resolution, NOIDA correctly diagnoses 96.0% of defects; in contrast, commercial diagnosis identifies 72.0% of defects accurately. On average, the physical resolution for NOIDA is 14.7, which is 3.4% better than commercial diagnosis. Besides reporting fewer candidates, NOIDA diagnoses 13.5% more defects correctly, on average. Additionally, for static (sequence-dependent) defects, the mean physical resolution of NOIDA is 11.0% (31.1%) better than commercial diagnosis. Thus, for primitive cells, Figures 5.11 and 5.13 reveal that NOIDA outperforms commercial diagnosis in terms of physical accuracy as well as resolution.

Figure 5.14 reveals that, for complex cells, NOIDA correctly diagnoses 74.0% of defects when the physical resolution is one, which is an enhancement of 32.1% over commercial

diagnosis. However, NOIDA appears to report more candidates than commercial diagnosis, on average. Although commercial diagnosis seems to achieve better average physical resolution than NOIDA, it does so by sacrificing accuracy. Specifically, among the cases where commercial diagnosis reports fewer candidates, 22.4% of them are inaccurate. On the other hand, among the cases where NOIDA reports fewer candidates, 91.4% of them are accurate.

One of the defect types that is often seen is a contact-to-gate short (i.e., a bridge defect between the drain/source and the gate of a transistor) [364]. Among primitive cells, it is observed that NOIDA diagnoses all such defects accurately, while commercial diagnosis is unable to identify the correct location for 5.1% of defects. Additionally, NOIDA hits a home run five times more often than commercial diagnosis. When contact-to-gate shorts are examined for complex cells, it is revealed that the average accuracy for NOIDA is 99.7%, which is 9.9% more than commercial diagnosis. Moreover, NOIDA delivers a home run 8X times commercial diagnosis.

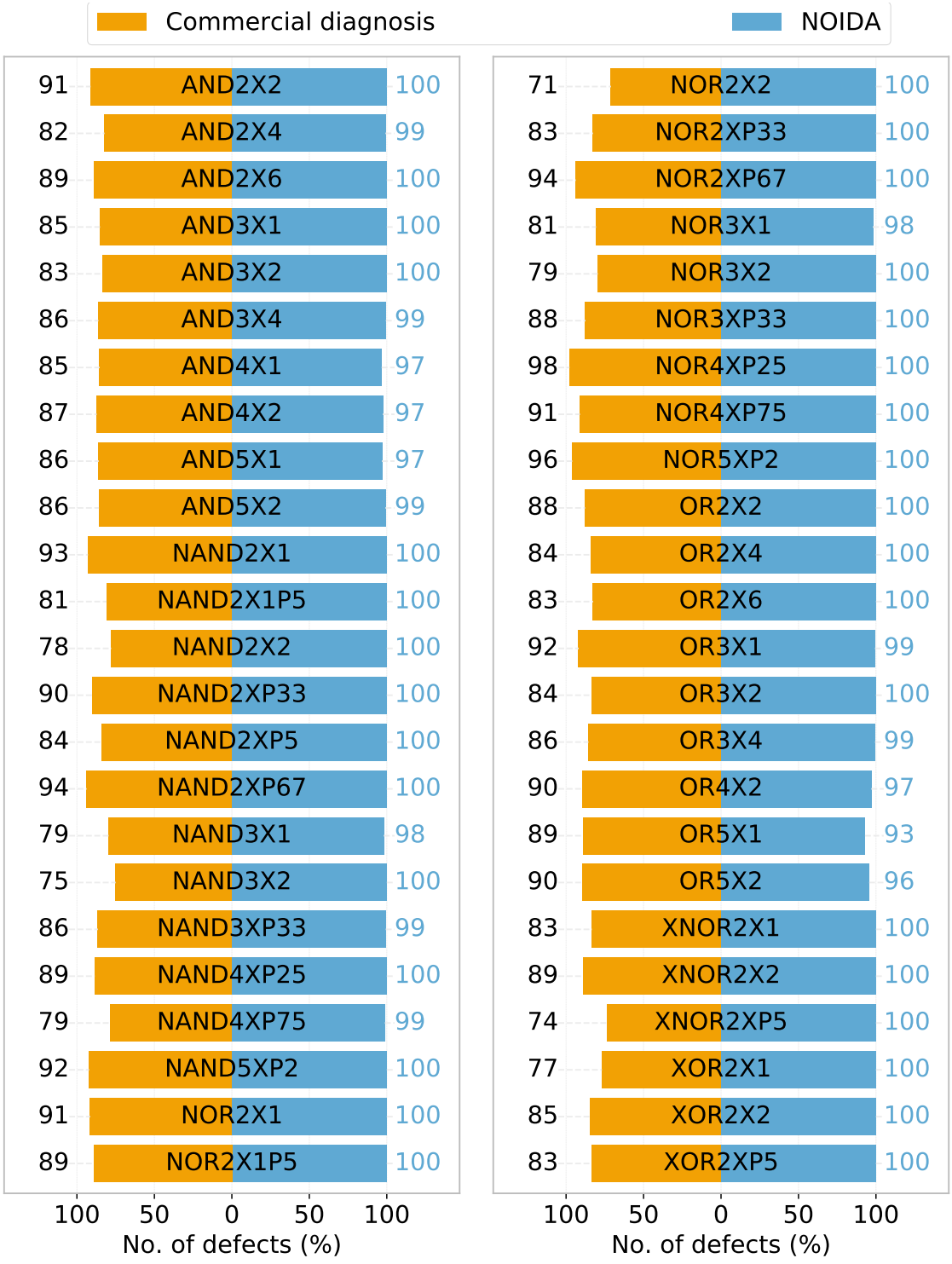


Figure 5.11: Physical accuracy for each primitive cell in a 7nm standard-cell library, when each cell is exhaustively tested.

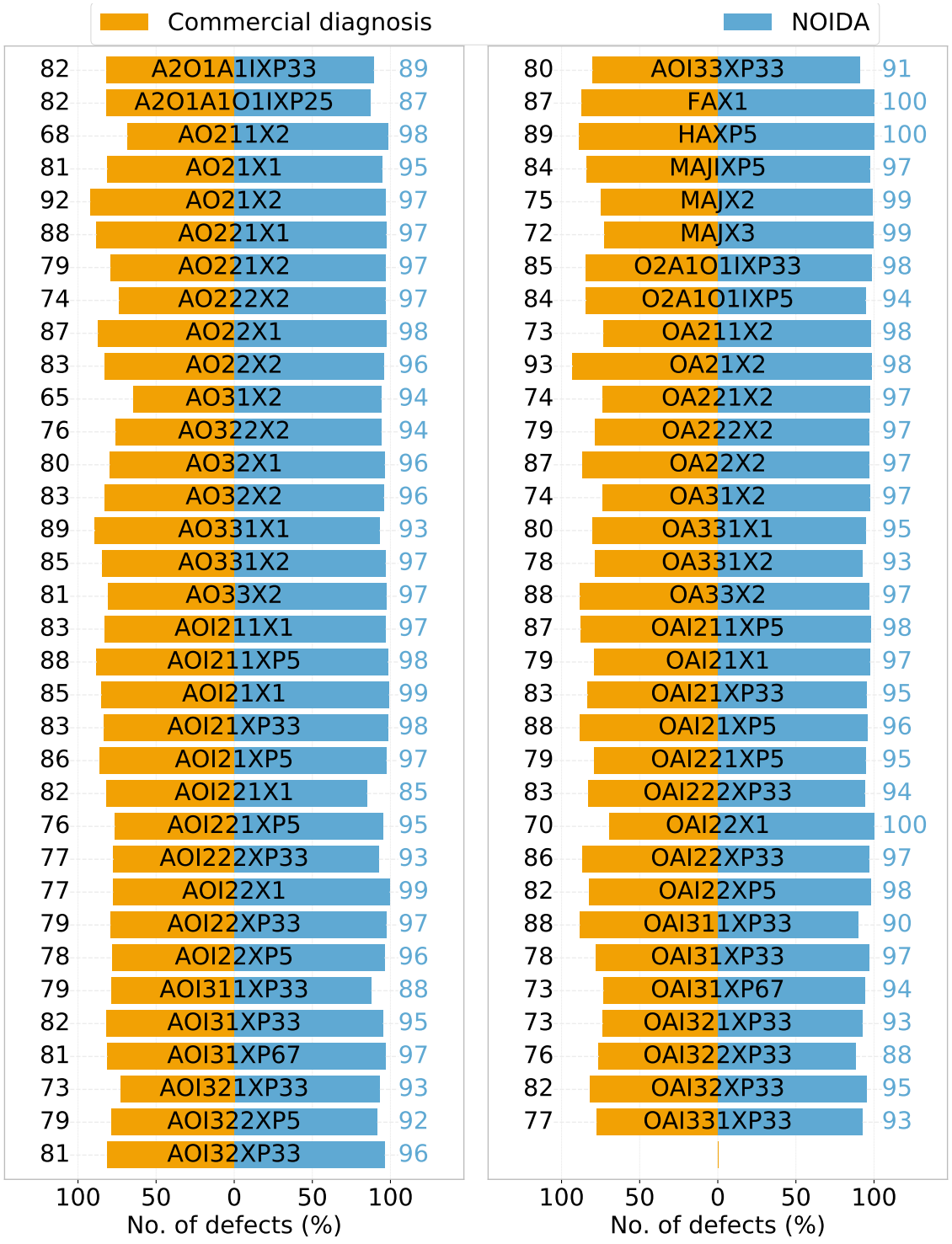


Figure 5.12: Physical accuracy for each complex cell in a 7nm standard-cell library, when each cell is exhaustively tested.

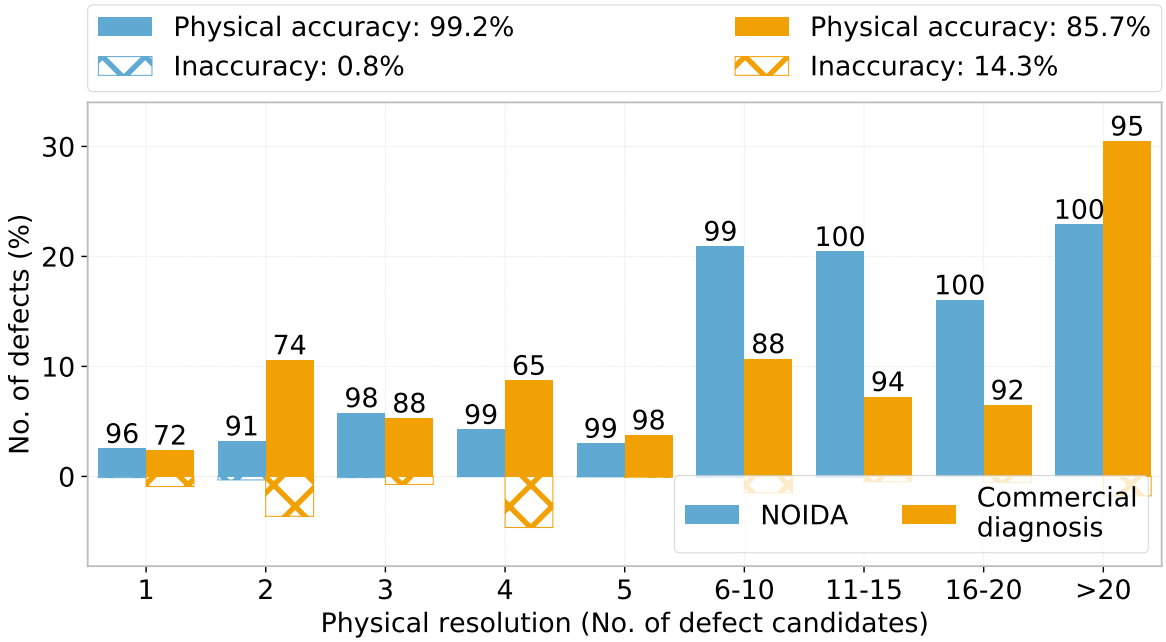


Figure 5.13: Distribution of physical resolution for primitive cells in a 7nm standard-cell library, when each cell is exhaustively tested.

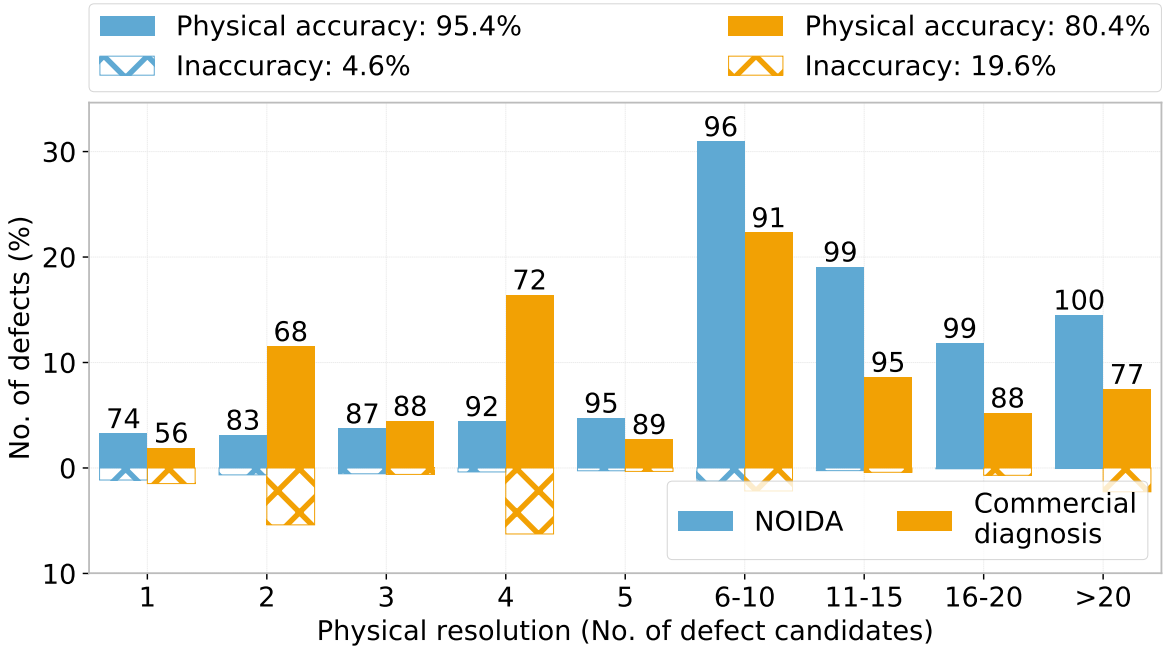


Figure 5.14: Distribution of physical resolution for complex cells in a 7nm standard-cell library, when each cell is exhaustively tested.

5.4.4 Results: Front-end ATPG

The experiment conducted in Section 5.4.3 assumes that each standard cell is exhaustively tested. However, one of the major problems with a cell-exhaustive test set [68–70, 86, 87] is the amount of test time and test data volume, which are directly proportional to the number of test patterns. Thus, a cell-exhaustive test set may not be suitable in practice.

In this section, NOIDA is applied to each defective cell while assuming that each cell is tested with a minimum number of test patterns required to detect all the defects modeled within a cell. A commercial ATPG tool that is capable of generating a test set of minimal size is employed here to create tests for each standard cell.

Figure 5.15 shows the size of the test set generated by commercial ATPG as a ratio to the number of exhaustive tests for each cell. The vertical axis shows the ratio of the test set sizes. The horizontal axis represents the standard cells. The solid blue line in Figure 5.15 shows the ratio of the test set sizes for static defects, while the dashed yellow line shows the ratio for sequence-dependent defects. It should be noted that the y-axis uses a log scale. Expectedly, ATPG generates fewer tests than exhaustive testing. For static defects, the minimum ratio for the test set sizes is 0.05; for sequence-dependent defects, the minimum ratio is 0.0004.

Figure 5.16 shows the average physical accuracy for each primitive cell. Figure 5.16 reveals that NOIDA performs better than state-of-the-art commercial diagnosis for every primitive cell. The physical accuracy achieved by NOIDA for each primitive cell is 100.0%, which is 14.2% higher than commercial diagnosis. Moreover, the maximum improvement in accuracy is observed for the cell, "NOR2X2", where commercial diagnosis reports a correct candidate for 29.0% fewer defects. Furthermore, it is observed that commercial diagnosis performs slightly worse for sequence-dependent defects than static defects. Specifically, for static (sequence-dependent) defects, NOIDA attains an improvement of 13.4% (16.0%) over commercial diagnosis.

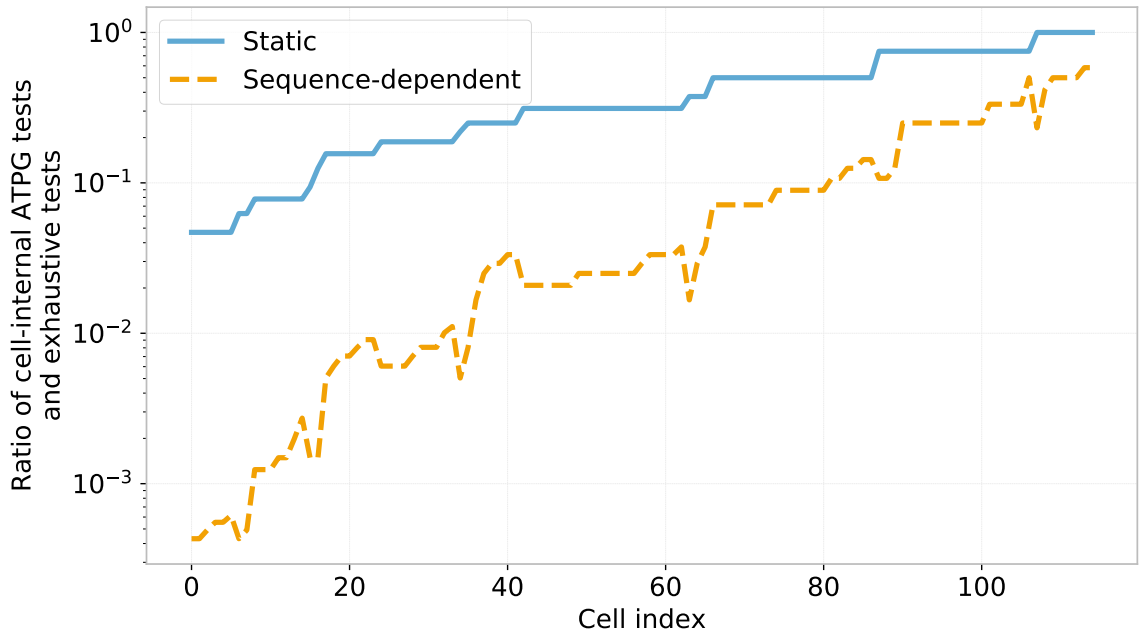


Figure 5.15: Ratio of test set sizes obtained from a commercial cell-internal ATPG tool and exhaustive testing for a 7nm standard-cell library.

Figure 5.17 shows the average physical accuracy for each complex cell. It is clearly seen from the figure that NOIDA outperforms commercial diagnosis for each complex cell as well. More importantly, the physical accuracy attained by NOIDA for each complex cell except one is perfect; for the cell named "OAI31XP33", the physical accuracy for NOIDA is 98.6%. On the other hand, the average physical accuracy attained by commercial diagnosis is 80.5%. In addition, the maximum improvement in physical accuracy is observed for the cell, "AO31X2", where the accuracy for commercial diagnosis is 35.0% less than NOIDA. Furthermore, it is observed that NOIDA achieves 100.0% (99.97%) accuracy for static (sequence-dependent) defects, which is an improvement of 22.3% (16.8%) over commercial diagnosis.

When the results presented in Figures 5.16 and 5.17 are compared with Figures 5.11 and 5.12, respectively, i.e., when the ramifications of using an ATPG test set instead of an exhaustive test set are studied, it is discovered that NOIDA performs better (in terms

of physical accuracy) with fewer patterns. It has been shown in [187–190] that certain tests can have a detrimental effect on the quality of back-end diagnosis; that observation is demonstrated here to be true for front-end diagnosis as well. A test can confound and, consequently, misdirect diagnosis, which could either deteriorate physical accuracy (if the correct candidate is removed) or physical resolution (if diagnosis yields more candidates) or both. This observation is especially true for multiple defects (or a single defect affecting multiple nets), where error masking and unmasking effects are prominent.

Figures 5.18 and 5.19 show histograms of the physical resolution for primitive and complex cells, respectively. It is observed from Figure 5.18 that among the defects diagnosed with perfect resolution, NOIDA correctly diagnoses each defect; in contrast, commercial diagnosis diagnoses 73.0% of defects accurately. On average, NOIDA reports 10.8 candidates per diagnosis, which is 39.8% better than commercial diagnosis. Besides reporting fewer candidates, NOIDA diagnoses 14.2% more defects correctly, on average. Thus, Figures 5.16 and 5.18 show that NOIDA outperforms commercial diagnosis in terms of physical accuracy as well as resolution for primitive cells.

Figure 5.19 reveals that, for complex cells, commercial diagnosis returns a correct candidate for 27.8% fewer defects, when the physical resolution is at most five. Additionally, NOIDA reports fewer candidates for 22.6% of defects without losing accuracy. However, NOIDA appears to report more candidates than commercial diagnosis, on average. Although commercial diagnosis seems to attain better average physical resolution than NOIDA, it does so by sacrificing accuracy. Specifically, among the cases where commercial diagnosis reports fewer candidates, 21.3% of them are inaccurate.

When the physical resolution achieved by NOIDA is compared for exhaustive and ATPG test sets, it is observed that the mean resolution is slightly better when an ATPG test set is used. Specifically, the improvement is 6.6%. However, more front-end defects are diagnosed with ideal resolution when an exhaustive test set is employed (specifically, 15X more often).

To summarize the results presented in this section, when each cell in the standard-cell library is tested with an ATPG test set, the physical accuracy for NOIDA is 100.0%, while the accuracy for commercial diagnosis is 82.1%. NOIDA improves physical resolution for 33.7% of defects without losing accuracy. In contrast, commercial diagnosis reports fewer but incorrect candidates for 21.4% of defects. Eliminating a correct defect candidate can have an adverse effect on subsequent failure analysis methods. For example, an incorrect root cause can be estimated for a population of failing chips due to an imprecise construction of defect paretos from analyzing numerous diagnoses, which could result in an unnecessary usage of PFA resources, thereby impeding yield learning.

Results presented in Sections 5.4.3 and 5.4.4 are compared to study the consequences of using an ATPG test set instead of an exhaustive test set for diagnosis. When an exhaustive test set is utilized by NOIDA, more front-end defects with an ideal resolution are diagnosed; on the other hand, the use of an ATPG test set results in better physical resolution and accuracy, on average.

When contact-to-gate shorts are analyzed, it is observed that, among primitive cells, NOIDA diagnoses all such defects accurately, while commercial diagnosis is unable to identify the correct location for 5.3% of defects. Additionally, NOIDA hits a home run twice more often than commercial diagnosis. When contact-to-gate shorts are examined for complex cells, it is revealed that the average accuracy for NOIDA is 100.0%, which is 9.8% more than commercial diagnosis. Moreover, NOIDA delivers a home run 1.7X times commercial diagnosis.

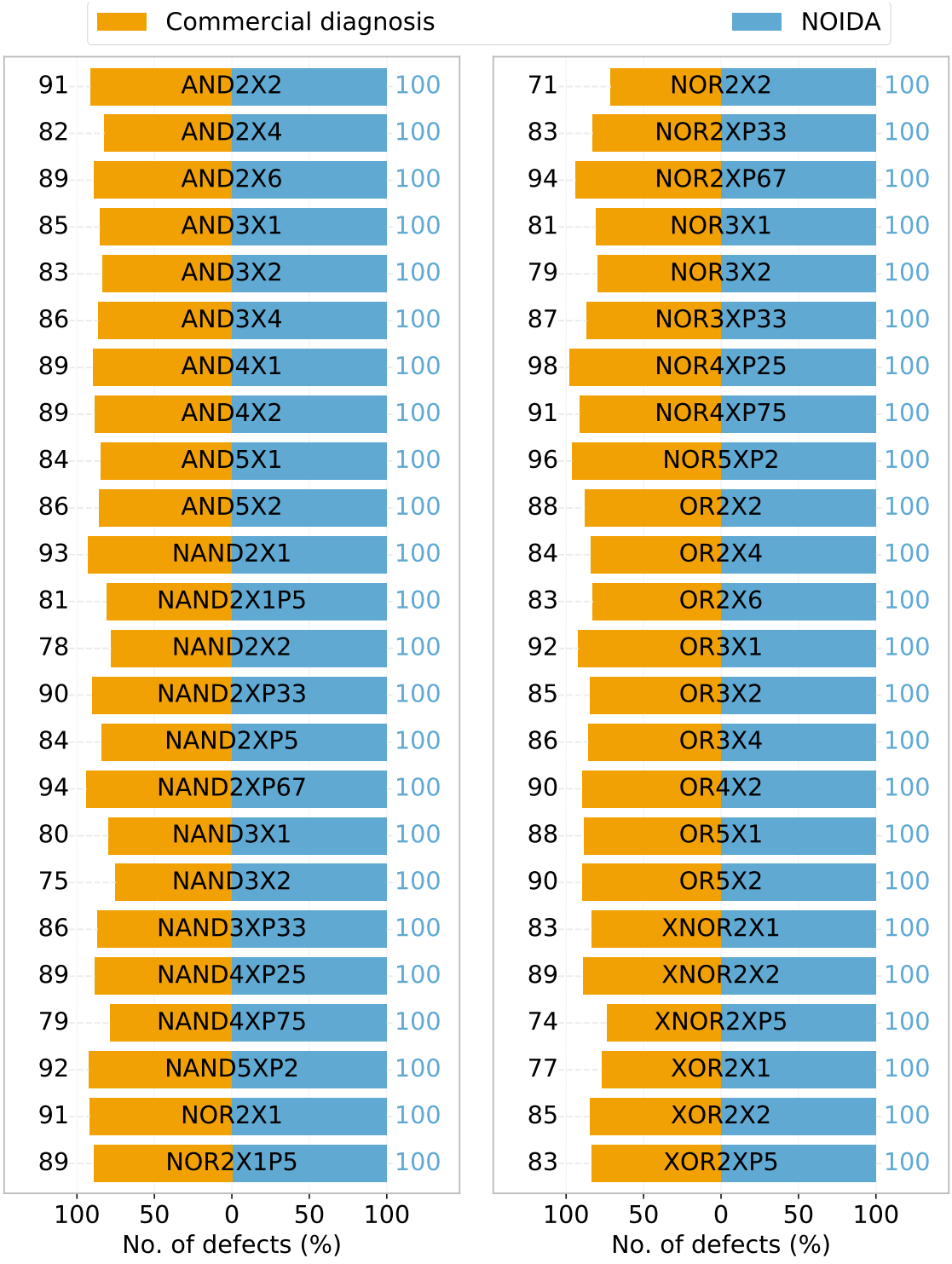


Figure 5.16: Physical accuracy for each primitive cell in a 7nm standard-cell library, when each cell is tested with an ATPG test set.

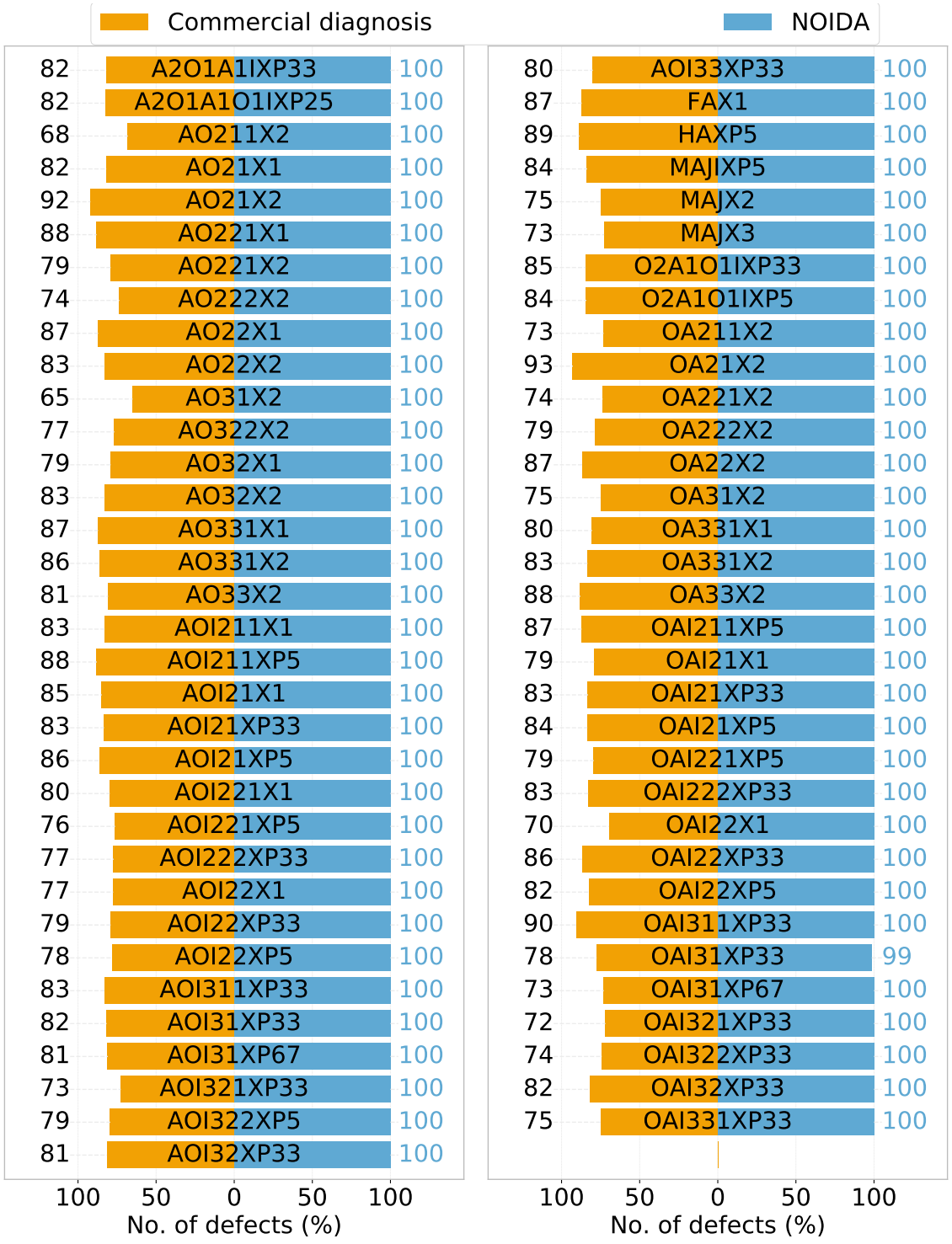


Figure 5.17: Physical accuracy for each complex cell in a 7nm standard-cell library, when each cell is tested with an ATPG test set.

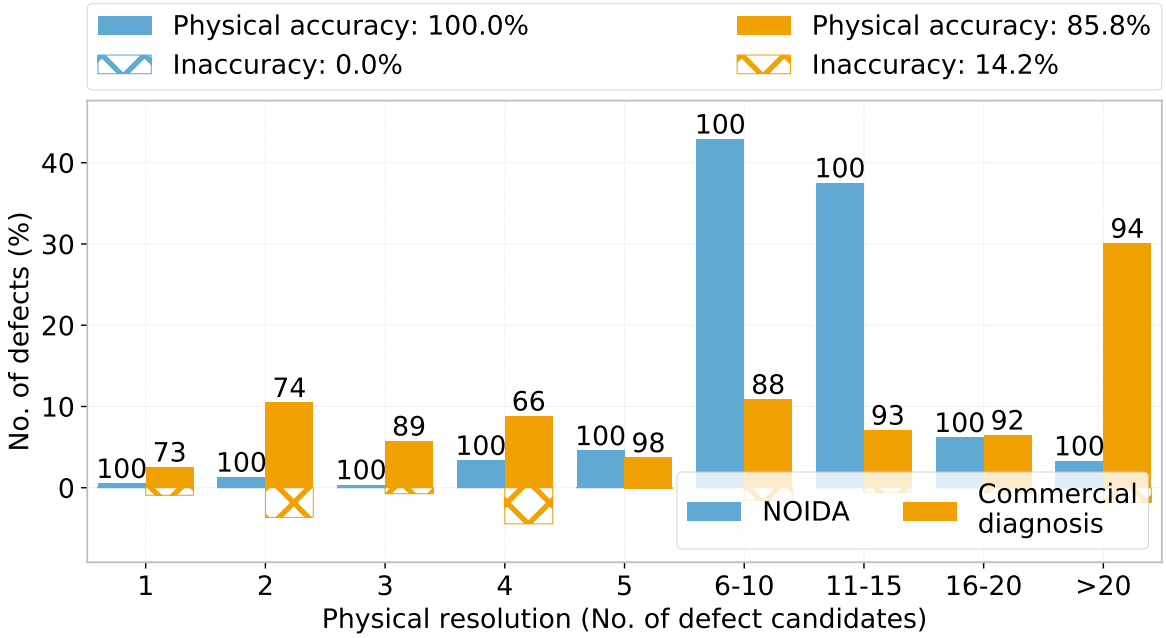


Figure 5.18: Distribution of physical resolution for primitive cells in a 7nm standard-cell library, when each cell is tested with an ATPG test set.

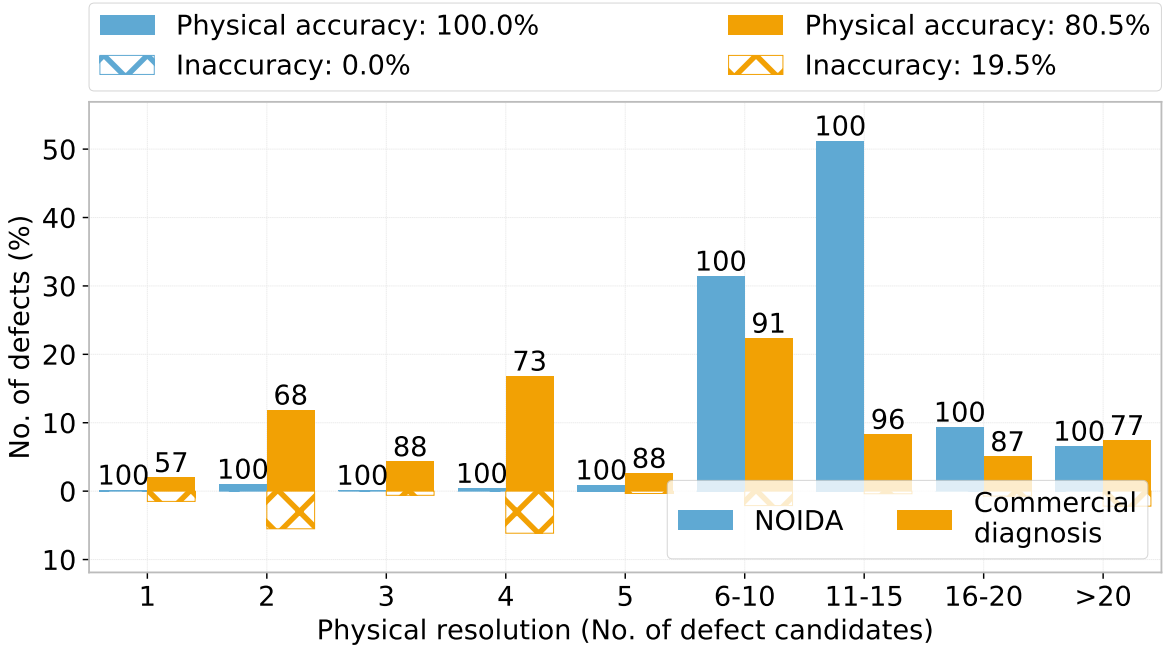


Figure 5.19: Distribution of physical resolution for complex cells in a 7nm standard-cell library, when each cell is tested with an ATPG test set.

5.4.5 Results: Circuit-level Noise

As discussed towards the end of Section 5.2, circuit-level noise can produce an unpredictable deviation between the tester response and the behavior estimated by a fault model. In NOIDA (and essentially any prior front-end diagnosis approach that focuses on physical localization), a deviation between the observed and the predicted behavior can be caused due to the potentially inexact SPICE models employed for transistor-level fault simulation. Voltage approximated at an intra-cell node from an analog simulation could be misinterpreted as logic-0 instead of logic-1 (or vice versa), which could result in a cell-level failing pattern be misconstrued as a cell-level passing pattern (or vice versa). In NOIDA, the possibly incorrect analog-to-digital conversion could lead to an imprecise derivation of the neighborhood state for an intra-cell node, and consequently, mistakenly, eliminate a consistent candidate or retain an inconsistent candidate.

An experiment is thus conducted to study the effect of noise on the performance of NOIDA and commercial diagnosis. The experiment is designed to mimic the undesired transformation of a cell-level passing pattern to a failing pattern (or vice versa) due to noise. Specifically, each front-end defect response generated in Section 5.4.2 is altered by randomly flipping the pass-fail behavior of a pattern such that at most 50% of the patterns are transformed.

Figure 5.20 shows the average physical accuracy for each primitive cell. Figure 5.20 reveals that NOIDA performs better than state-of-the-art commercial diagnosis for every primitive cell when noise is introduced in the tester response. On average, the physical accuracy for NOIDA is 95.8%, which is 3.5X times better than commercial diagnosis. Out of 48 primitive cells, 24 (50.0%) cells achieve perfect physical accuracy. Moreover, the maximum improvement in accuracy is observed for the cell, “NAND4XP25”; while NOIDA correctly diagnoses 96.0% of defects, commercial diagnosis correctly locates a mere 16.0% of defects. Further investigation reveals that NOIDA achieves 97.8% (93.6%) accuracy for

static (sequence-dependent) defects, which is 4.8X (2.7X) times commercial diagnosis. Additionally, NOIDA diagnoses each static defect correctly for 39 (81.3%) primitive cells.

Figure 5.21 shows the average physical accuracy for each complex cell. It is seen that NOIDA outperforms leading-edge commercial diagnosis for each complex cell in the presence of noise as well. Although the effect of noise on diagnosis quality is more prominent for complex cells, NOIDA locates more defects correctly than commercial diagnosis. On average, the physical accuracy for NOIDA is 74.0%, which is 32.0% higher than commercial diagnosis. Out of 67 complex cells, one cell ("HAXP5") achieves perfect physical accuracy when NOIDA is used; on the other hand, the maximum physical accuracy attained by commercial diagnosis is only 60.9%. In addition, the maximum improvement in physical accuracy is observed for the cell, "OAI31XP67", where the accuracy is 4.5X times commercial diagnosis. Further exploration reveals that NOIDA is more effective for static than sequence-dependent defects. For static defects, NOIDA attains 87.7% accuracy, which is 3X times commercial diagnosis. Moreover, NOIDA identifies the correct candidate for each static defect for 11 (16.4%) primitive cells. For sequence-dependent defects, NOIDA achieves 64.8% accuracy, an improvement of 47.3% over commercial diagnosis.

Figures 5.22 and 5.23 compare the physical resolution achieved by NOIDA and commercial diagnosis in the presence of noise for primitive and complex cells, respectively. It is observed from Figure 5.22 that, for primitive cells, NOIDA achieves perfect resolution for 14.1% of defects, of which 90.2% are accurate. In contrast, commercial diagnosis reports a resolution of one for 8.6% of defects with an accuracy of 25.6%. NOIDA reports a single correct candidate for 12.8% of defects, which is 5.7X more often than commercial diagnosis.

When the effect of noise on the quality of diagnosis attained by NOIDA for primitive cells is investigated (i.e., when Figures 5.18 and 5.22 are compared), it is observed that the accuracy decreases by 4.2%, but the number of ideal diagnoses increases from 0.5% to 12.8%. Thus, the improvement in resolution also degrades the achieved accuracy. On the

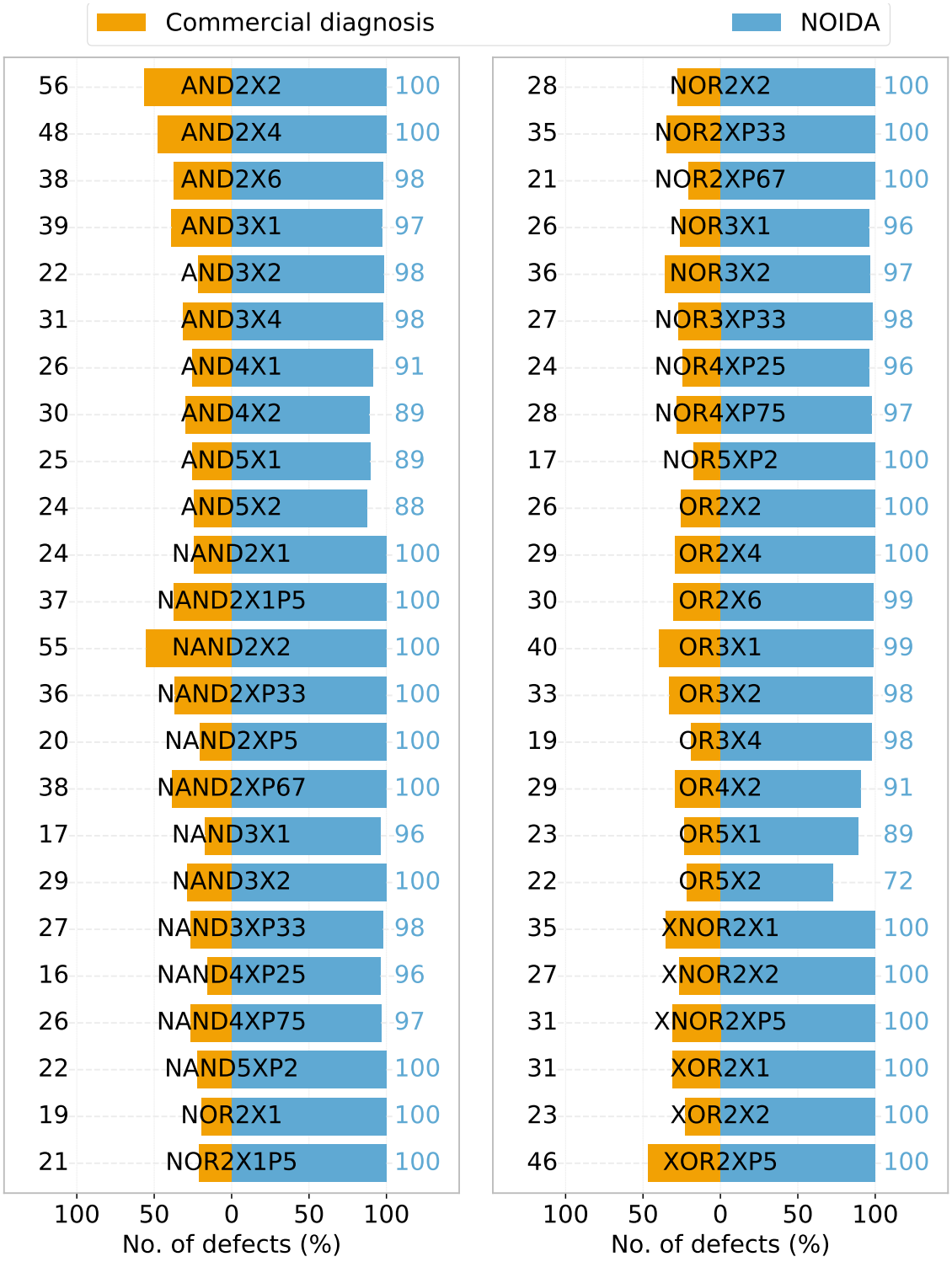


Figure 5.20: Physical accuracy for each primitive cell in a 7nm standard-cell library in the presence of circuit-level noise.

other hand, the accuracy attained by commercial diagnosis drops from 85.8% to 27.4% in the presence of noise.

Figure 5.23 reveals that, for complex cells, NOIDA achieves perfect resolution for 24.9% of defects, of which 53.6% are accurate. In contrast, commercial diagnosis reports a single candidate for 5.6% of defects with an accuracy of 28.4%. Moreover, NOIDA reports 6.3 candidates per diagnosis, on average, which is 2.9% better than commercial diagnosis. Thus, NOIDA outperforms commercial diagnosis in terms of physical accuracy as well as resolution for complex cells, when noise is introduced in the tester response.

Conceivably, the effectiveness of NOIDA decreases in the presence of noise for complex cells (Figures 5.19 and 5.23). Specifically, although NOIDA reports almost half as many candidates on average, its physical accuracy decreases from 100.0% to 74.0% because of noise. The accuracy degradation is, however, greater for commercial diagnosis, where the attained accuracy plummets to 42.0%.

To summarize the results presented in this section, in the presence of noise, the mean physical accuracy for NOIDA is 78.5%, while the accuracy for commercial diagnosis is 39.0%. NOIDA reports 0.9 more candidates per diagnosis, on average, while identifying the correct candidate twice as often as commercial diagnosis.

When the effect of noise is investigated for contact-to-gate shorts, it is observed that, among primitive cells, the average accuracy for NOIDA is 93.3%, while commercial diagnosis identifies the correct location for only 21.8% of defects. Additionally, NOIDA hits a home run 7.6X times commercial diagnosis. When contact-to-gate shorts are analyzed for complex cells, it is revealed that the average accuracy for NOIDA is 75.4%, which is 2.6X times commercial diagnosis. Moreover, NOIDA delivers a home run 16.1X more often than commercial diagnosis.

Results presented in Section 5.4.4 and 5.4.5 are compared to study the consequences of introducing noise to the tester response. The results demonstrate that NOIDA is more

resistant to noise than state-of-art commercial diagnosis. Specifically, while noise decreases the average physical accuracy for NOIDA by 21.5%, noise drastically worsens the accuracy for commercial diagnosis from 82.3% to 39.0%. However, noise seemingly improves resolution for each diagnosis approach. Specifically, NOIDA reports 4.2 fewer candidates per diagnosis, on average, while commercial diagnosis reports 3.3 fewer candidates because of noise.

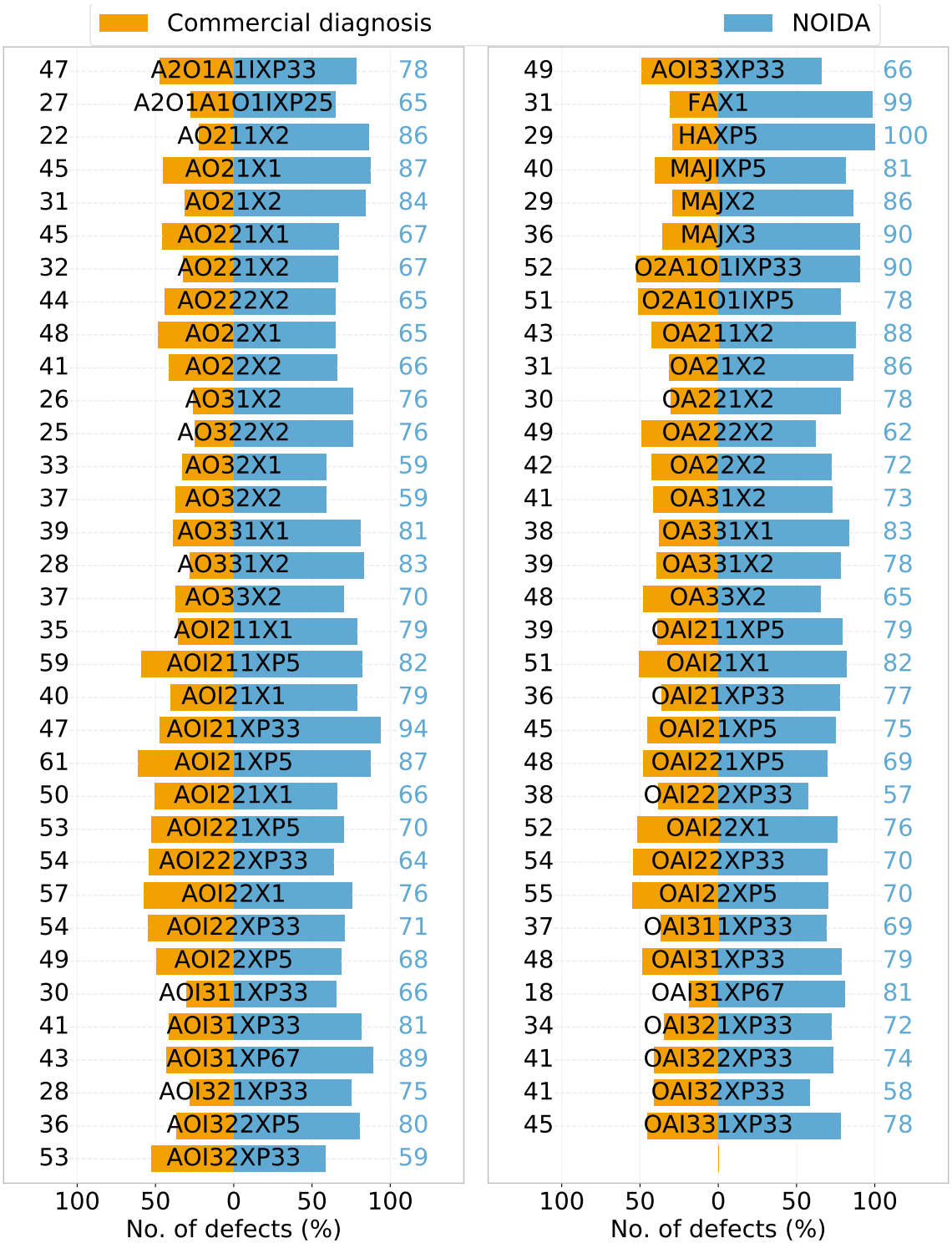


Figure 5.21: Physical accuracy for each complex cell in a 7nm standard-cell library in the presence of circuit-level noise.

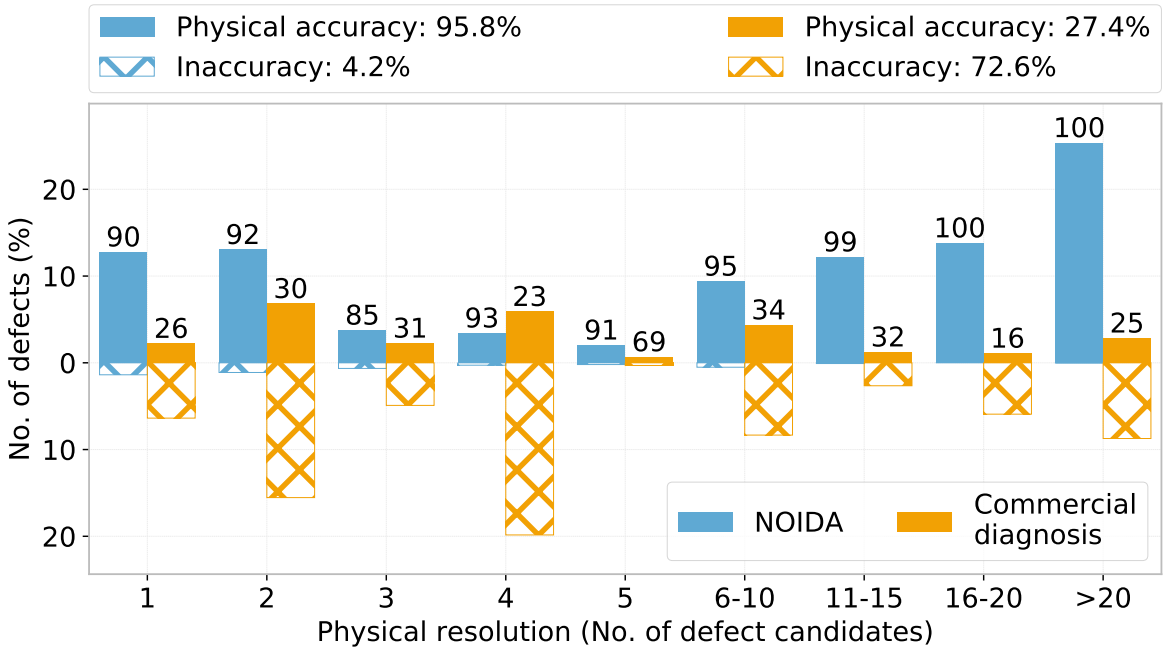


Figure 5.22: Distribution of physical resolution for primitive cells in a 7nm standard-cell library in the presence of circuit-level noise.

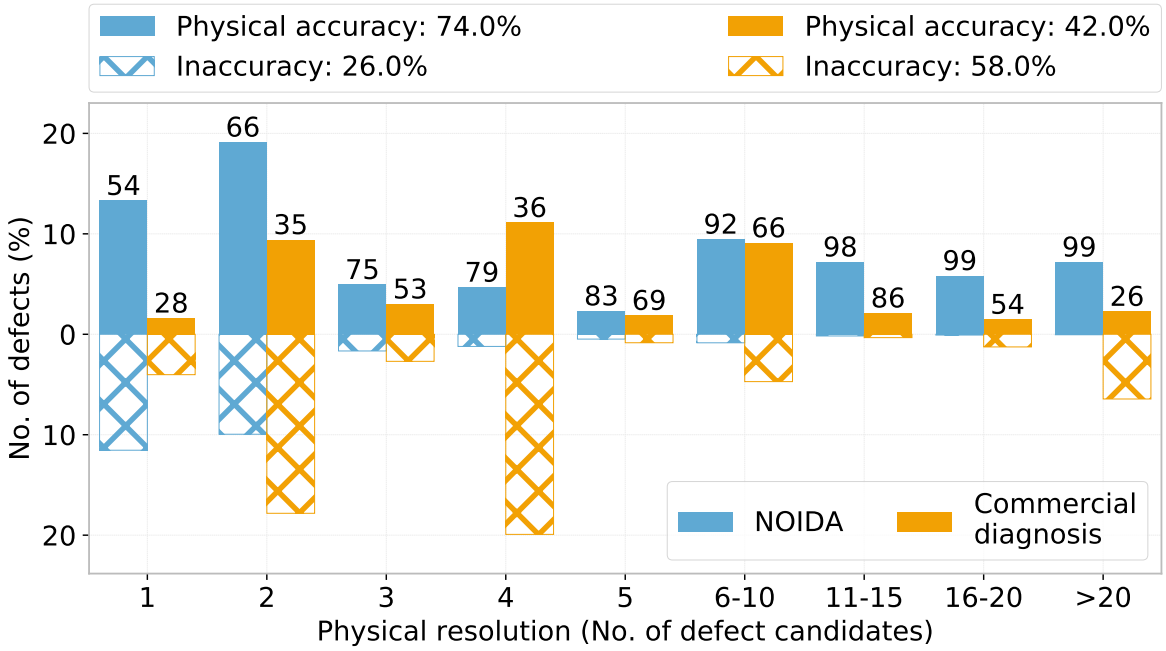


Figure 5.23: Distribution of physical resolution for complex cells in a 7nm standard-cell library in the presence of circuit-level noise.

5.5 Conclusion

This chapter describes a novel generalized methodology for front-end defect diagnosis we call NOIDA (NOise-resistant Intra-cell Diagnosis Approach). NOIDA concentrates on finding defect locations within a cell and deriving defect behavior based on the nets that surround the suspected defect location. Unlike prior work where a fault model is employed to verify if the assumed model represents an actual defect, NOIDA extracts a custom fault model of a defect by examining its physical neighborhood. Thus, in addition to physically localizing a front-end defect (in terms of its $x - y - z$ location), NOIDA characterizes the defect with respect to its behavior as well.

A comprehensive simulation experiment is conducted to study the effectiveness of NOIDA. NOIDA is evaluated on over 34,000 front-end defects that are distributed over 115 standard cells within a 7nm standard-cell library. The performance of NOIDA is investigated under three scenarios: (a) each defective cell is exhaustively tested; (b) each defective cell is tested with an ATPG test set; and (c) each defective cell is tested in the presence of circuit-level noise. Experiment results demonstrate the efficacy of NOIDA.

In the first scenario (i.e., when an exhaustive test set is applied to a defective standard cell), NOIDA accurately diagnoses 96.6% of defects, which is an improvement of 17.7% over state-of-the-art commercial diagnosis. In addition, NOIDA reports an ideal diagnosis (i.e., when a single candidate reported by diagnosis is correct) for 50.5% more defects.

In the second scenario, i.e., when each defective cell is tested with a test set generated by a commercial ATPG tool, it is observed that the average physical accuracy for NOIDA is 100.0%, which is 21.8% better than commercial diagnosis. Additionally, the physical resolution for NOIDA is better than commercial diagnosis for 33.7% of defects. Although commercial diagnosis reports 15.0% fewer candidates, its low accuracy can significantly impact subsequent failure analyses. For example, a statistical analysis of low accuracy diagnoses

can lead to an incorrect prediction of the underlying root cause, which, in turn, steers the slow and destructive PFA process in the wrong direction.

Compared to an exhaustive test set, the physical accuracy for NOIDA is discovered to be 3.4% higher when an ATPG test set is utilized. This result can be explained from the observation that certain tests, especially in the presence of multiple faults, can confound diagnosis due to error masking and unmasking. On the other hand, it is observed that the likelihood of producing an ideal diagnosis for NOIDA decreases when an ATPG test set is used. This is because additional tests can distinguish among existing candidates, which results in improved resolution for an exhaustive test set. Therefore, efforts should be directed towards generating an optimal, diagnostic test set that can improve physical resolution without sacrificing physical accuracy.

In the third scenario, the effect of adding noise to the tester response is analyzed. Various factors can cause deviations between the behavior determined by a fault model and the behavior observed on the tester. Approximate SPICE models that are employed to perform analog simulation can induce an undesired discrepancy between the actual and the predicted behavior of a failing standard cell, which can cause diagnosis to misinterpret the logic value at a cell output (and/or any of the cell-internal nodes). As a consequence, a failing pattern can be misinterpreted as a passing pattern or vice versa, and lead diagnosis astray. Experiment results indicate that NOIDA is more resistant to noise than a state-of-the-art commercial tool. NOIDA correctly diagnoses 78.5% of the front-end defects, which is more than twice as often as commercial diagnosis. NOIDA returns an ideal diagnosis for 13.2% of defects, which is 7.6X times commercial diagnosis.

The superior quality of diagnosis achieved by NOIDA to localize and characterize the behavior of a front-end defect can make PFA efficient and cost-effective, which likely enhances/accelerates yield learning. However, there is still room for improvement. A machine learning based scoring framework, similar to that developed for LearnX and MD-LearnX,

can be adapted here to pinpoint the correct candidate. Layout features can be incorporated in the scoring model for improved performance. Future work is further discussed in detail in the next chapter.

Chapter 6

Conclusions

Yield, which is defined as the proportion of working chips fabricated, can be quite low when a new manufacturing process or a new chip design is introduced. The process of identifying and rectifying the sources of yield loss to improve both chip design and manufacturing is called yield learning.

As semiconductor fabrication advances to smaller and more complex process nodes, it is becoming increasingly difficult to ramp logic yield quickly. Advanced nodes introduce new design and manufacturing challenges, especially in logic, resulting in an increase in the number and complexity of defects, which consequently hinder yield learning. The rate of yield learning is extremely critical to the success of the semiconductor industry and must thus be accelerated to meet the three objectives of aggressive time-to-volume, time-to-market and time-to-money requirements.

Software-based defect diagnosis plays an important role in logic failure analysis and yield learning. Diagnosis is a process to identify the location and ideally, characterize the nature and root cause of a defective chip by examining its tester response. Based on the feedback produced by diagnosis, a small but significant number of chips are selected to be inspected physically. The aim of physical failure analysis (PFA) is to provide crucial understanding of

failure mechanisms to improve the design and/or the manufacturing process for increasing yield.

Besides guiding PFA, diagnosis offers valuable information to assist yield analysis and learning. Diagnosis aids in the (a) derivation of defect density and size distributions for each metal layer to comprehend the impact of random defects, (b) identification of yield-limiting defects by statistically correlating a population of failing chips, (c) evaluation of the capability of various fault models, (d) generation of an adaptive test set to curtail test escapes, and (e) prediction of the number of defective chips that will pass testing to estimate test cost.

Logic diagnosis is thus indispensable to the improvement of the design and manufacturing of a chip. It plays a significant role during (a) yield ramping, when logic test chips are employed and customer chips are fabricated ahead of time to meet aggressive market demands, (b) high-volume manufacturing for continuous yield monitoring, and (c) yield excursion, when an underlying source of yield loss needs to be identified to stabilize yield.

With increasing chip density and manufacturing complexity, and decreasing critical defect size, the effectiveness of PFA in characterizing the root cause of a defect is diminishing. On the other hand, contrary to PFA, diagnosis is expeditious and non-destructive in nature; in addition, it is influential in guiding PFA and driving yield learning, in general. The benefits of diagnosis, along with the declining efficiency of PFA, thus makes advancing diagnosis procedures a way to enable/accelerate yield learning. The logic diagnosis methodology developed in this dissertation is an important step towards realizing that goal. The rest of this chapter encapsulates the contributions of our work presented in this dissertation, and provides some important areas of future work.

6.1 Dissertation Contributions

Given the advantages of diagnosis and the limitations of PFA, it is crucial that the responsibility of characterizing a defect, which is traditionally accomplished by PFA, is fulfilled by diagnosis as much as possible to climb the yield learning curve faster. Therefore, the objective of diagnosis must be to logically characterize a defect. In other words, diagnosis must concentrate on accurately pinpointing the $x-y-z$ location of a defect in the circuit as well as determine the precise logic behavior of a defect. An ideal diagnosis thus, minimizes the need for PFA.

A comprehensive diagnosis methodology is developed in this dissertation that endeavors to actualize the aforementioned objectives. Our developed diagnosis methodology is divided into three stages. The first stage focuses on defect localization, while the second and the third stages concentrate on deducing the behavior of a back-end and a front-end defect, respectively, while further improving its physical location. The significance and the major contributions of this dissertation are summarized next.

6.1.1 LearnX/MD-LearnX

LearnX/MD-LearnX is a physically-aware diagnosis methodology that identifies the location and the type of the defect affecting a failing chip [274, 275]. LearnX assumes that a single defect resides in a failing chip. MD-LearnX builds on LearnX to effectively locate multiple defects. LearnX/MD-LearnX serves as the first stage of our diagnosis methodology and impacts the effectiveness of subsequent analyses. Notable features of LearnX/MD-LearnX include:

- **Use of the X-fault model [65] for improved diagnostic accuracy**: Besides utilizing the commonly used (temporary) stuck-at fault model to represent an error at a likely defect location, the X-fault model is employed to enhance diagnosis accuracy.

The X-fault model assumes an unknown value (X) at a candidate location, thereby, allowing the error to propagate conservatively, even for multiple defects.

- **Machine learning for improved diagnostic resolution**: In prior work, a candidate-ranking method is manually designed from domain knowledge and/or intuition. However, its effectiveness is not guaranteed for every defect type, design, and/or manufacturing process. Instead, in our work, a data-driven candidate-scoring model is automatically created from learning the latent correlations between the tester response and simulation of the correct candidate.

- **Use of physical layout for improved diagnosis quality**: Layout information is utilized in our work to identify likely defect locations, filter out bridges that are physically infeasible, distinguish between an open and a dominant bridge defect, and identify the physical fan-out branches that may be faulty based on the pass-fail status of the receiver standard cells.

- **Outperform state-of-the-art**: Results from a simulation-based experiment demonstrate that LearnX returns a single correct candidate (i.e., an ideal diagnosis) for 67.9% fail logs, which is 67.5% more than state-of-the-art commercial diagnosis. In addition, for multiple defects, MD-LearnX reports an ideal diagnosis (i.e., a single correct candidate for each injected defect) for 40.0% of the fail logs, which is twice as often as commercial diagnosis.

  A silicon experiment further demonstrates the effectiveness of LearnX/MD-LearnX. The silicon failure data comes from a 40M gate test chip design manufactured in a 14nm technology. When 36 failing chips are physically inspected, it is observed that LearnX/MD-LearnX returns fewer candidates than commercial diagnosis for 69.4% chips, while identifying the correct candidate in each case.

6.1.2 PADLOC

The goal of PADLOC is to characterize a back-end defect in terms of its exact physical location and precise logic behavior [146]. PADLOC further localizes a defect by identifying the net segments that are relevant for defect excitation. It derives the logic behavior of a defect by examining the logic activity of the adjacent segments. Salient features of PADLOC include:

- **Net segmentation for enhanced physical localization**: A candidate net is divided into smaller segments based on its topology and the topology of its physical neighbors. A two-line bridge defect, for example, likely resides between two adjacent segments that have an unobstructed line-of-sight between them.

- **Avoid the use of imprecise circuit parameters**: Prior work utilizes circuit parameters such as coupling capacitances between adjacent segments, and logic threshold voltage and leakage current of a receiver cell to estimate the voltage at a candidate location. However, the potentially inaccurate values of the circuit parameters can lead diagnosis astray. Instead, the logic values established in the neighborhood of a candidate are analyzed to find the logic value at the candidate location.

- **Not limited to specific fault models**: Most work focuses on matching the observed circuit behavior with a fault model (e.g., the wired-bridge fault model, biased voting bridge fault model) to characterize the behavior of a defect. PADLOC instead deduces the precise logic behavior of a defect based on its neighborhood, and thus can diagnose a defect that portrays an arbitrary misbehavior.

- **Outperform state-of-the-art**: Results from a simulation-based experiment demonstrate that PADLOC correctly locates the defect for 96.5% fail logs, which is 10.3% more than leading-edge commercial diagnosis. *Bounding circle diameter*, which is de-

fined as the diameter of the smallest circle that encloses each segment suspected by diagnosis, is one of the metrics employed in this work to measure the physical resolution of a diagnosis approach. The value of *bounding circle diameter* is directly correlated to the cost of PFA [161]. It is observed that the average *bounding circle diameter* for PADLOC is 33.0% less (and hence, better) than commercial diagnosis.

A silicon experiment corroborates the capability of PADLOC. It is revealed that *bounding circle diameter* for PADLOC is 32.7% better than commercial diagnosis. More importantly, when 36 fail logs whose PFA results are available are examined, PADLOC reports a smaller *bounding circle diameter* for 47.2% fail logs (while reporting the same diameter for the remaining fail logs), and attains up to 44X improvement, without losing accuracy.

6.1.3 NOIDA

NOIDA characterizes a front-end defect in terms of its precise logic behavior and exact physical location within a failing standard cell [162]. NOIDA investigates the behavior of a candidate cell (implicated by LearnX/MD-LearnX) to pinpoint the location and derive the behavior of an intra-cell defect. Notable features of NOIDA include:

- **Fault dictionary free**: Prior work either creates a new fault dictionary or employs an existing one to identify a front-end (intra-cell) defect. The disadvantage of searching the observed cell behavior in the fault dictionary is that a dictionary-based approach, by design, cannot identify an unfamiliar defect. NOIDA, on the other hand, deduces the intra-cell candidates that could explain the observed behavior. In addition, it derives the logic behavior of a cell-internal defect by inspecting its intra-cell physical neighborhood.

- **Applicable to new and/or multiple defects**: Because NOIDA is independent

of a fault dictionary, it is capable of locating a new intra-cell defect that may be introduced by a new technology. In addition, prior fault dictionary-based approaches typically compute and store the responses of single cell-internal defects. NOIDA, on the other hand, makes no assumptions regarding the number of defects that could affect a standard cell.

- **Outperform state-of-the-art**: NOIDA is assessed on 115 cells within a 7nm standard-cell library [373]. Results from the first simulation-based experiment, where each defective cell is exhaustively tested, indicate that NOIDA correctly locates 96.6% of defects, which is 17.7% more than leading-edge commercial diagnosis. Moreover, NOIDA returns a single correct candidate for 50.5% additional defects.

  In the second experiment, where each defective cell is tested with a test set created with a commercial front-end ATPG tool, it is observed that NOIDA correctly identifies 21.8% more defects, and reports a single correct candidate for 38.0% more fail logs.

- **Resistant to circuit-level noise**: Approximate SPICE models employed during transistor-level simulation can cause an unforeseen deviation between the predicted and the observed defect behavior. For example, a cell-level failing pattern can be misconstrued as a passing pattern or vice versa. An experiment is designed to study the effect of introducing noise in the tester response on NOIDA and state-of-the-art commercial diagnosis. Results reveal that NOIDA is more robust to noise. Specifically, the accuracy for NOIDA is 78.5%, which is more than 2X times commercial diagnosis. Additionally, NOIDA returns an ideal diagnosis 7.6X more often than commercial diagnosis.

6.2 Future Work

A comprehensive diagnosis methodology is presented in this dissertation that outperforms leading-edge commercial diagnosis, thereby, facilitating yield learning. Some possible areas of research that should be investigated to further the work developed in this dissertation are described next.

6.2.1 Delay Defects

The scope of LearnX/MD-LearnX and PADLOC has been limited to diagnosing static defects, i.e., defects that require a single test pattern for detection and do not impact the circuit timing. Defects such as weak resistive opens and bridges may only modify the performance of a circuit instead of altering its logic functionality. Delay defects can be categorized based on location (spot or distributed delay) and the amount of delay (gross delay or small delay).

A spot defect occurs at a fixed location in the layout (e.g., opens and shorts). A defect whose delay is distributed over more than one circuit element is referred to as a distributed delay defect. A gross delay defect causes a delay sufficiently large enough to slow down a transition at a defect location such that it can be detected irrespective of the propagation path. A small delay defect, on the other hand, can only be detected if the error propagates through a path whose slack is less than its delay size. A delay defect requires at least two patterns for detection. A delay-defect diagnosis framework can be based on extending LearnX/MD-LearnX:

- A spot defect with gross delay can be detected by a sequence of two patterns, where the first pattern initializes the candidate location while the second pattern causes a transition at the location. Thus, for a two-pattern test, candidates can first be identified (by LearnX/MD-LearnX, for example) using the second pattern; a candidate can then be eliminated by verifying if it transitions between the two patterns.

- A small delay defect may not mirror the behavior of a stuck-at fault under the influence of the second pattern. A stuck-at fault at the defect location may produce more simulation-failing outputs than tester-failing outputs. Thus, the criterion of a stuck-at fault explaining a failing pattern can be relaxed to ignore TPSF outputs. Features pertaining to TPSF outputs can be ignored for small delay defect candidates during machine learning training/testing in LearnX/MD-LearnX.

- Diagnosis of distributed defects is more complicated but can be achieved through the use of path-delay, segment-delay and segment-network fault models [374].

- In aforementioned cases, PADLOC can be extended to diagnose a delay defect by monitoring the neighborhood state for a candidate for a sequence of patterns instead of a single pattern.

6.2.2 Design Features in LearnX/MD-LearnX

The features used to create a scoring model in LearnX (Phase 2) (and Phases 2 and 3 of MD-LearnX) are all derived by comparing the candidate simulation response and the tester response. Besides extracting features from the test data, design information can be used to identify new features for improved diagnosis quality. The work of [301], for instance, extracts design features to distinguish a bridge defect. For example, one feature checks if a pair of bridged nets drives the same standard cell; a bridge defect is that case cannot be differentiated from a cell defect.

Although those features do not particularly help in enhancing diagnostic resolution, they are used to convey information of the likelihood of a candidate being a bridge defect (or a non-bridge defect), which consequently aids in deriving defect type distributions during volume diagnosis.

6.2.3 Machine Learning for Ranking PADLOC and NOIDA Candidates

It is argued in Section 4.3.2.2 and Section 5.3 that the size of the neighborhood and the amount of inconsistency are the primary reasons why a correct candidate can be eliminated by PADLOC and NOIDA, respectively. One of the ways to improve diagnostic accuracy is to design a scoring method to rank the candidates analyzed by PADLOC and NOIDA. A procedure similar to LearnX/MD-LearnX, where machine learning is deployed to find the correct candidate, can be adopted in PADLOC and NOIDA. Features such as the number of physical neighbors, distance between adjacent segments and the number of (unique) failing/passing neighborhood states can be extracted for each candidate.

Bibliography

[1] G. E. Moore, "Cramming More Components onto Integrated Circuits," *Electronics*, vol. 38, no. 8, p. 114 ff, Apr 1965.

[2] G. E. Moore, "Progress in Digital Integrated Electronics," in *IEEE International Electron Devices Meeting (IEDM)*, vol. 21, 1975, pp. 11–13.

[3] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of Ion-implanted MOSFET's with Very Small Physical Dimensions," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 9, no. 5, pp. 256–268, 1974.

[4] P. S. Peercy, "The Drive to Miniaturization," *Nature*, vol. 406, no. 6799, p. 1023, 2000.

[5] C. A. Mack, "Fifty Years of Moore's Law," *IEEE Transactions on Semiconductor Manufacturing*, vol. 24, no. 2, pp. 202–207, 2011.

[6] R. R. Schaller, "Moore's Law: Past, Present and Future," *IEEE Spectrum*, vol. 34, no. 6, pp. 52–59, June 1997.

[7] M. Lundstrom, "Moore's Law Forever?" *Science*, vol. 299, no. 5604, pp. 210–211, 2003.

[8] S. Salahuddin, K. Ni, and S. Datta, "The Era of Hyper-scaling in Electronics," *Nature Electronics*, vol. 1, no. 8, pp. 442–450, 2018.

[9] B. Lojek, *History of Semiconductor Engineering.* Springer, 2007.

[10] W. Regitz and J. Karp, "A Three Transistor-cell, 1024-bit, 500 NS MOS RAM," in *IEEE International Solid-State Circuits Conference (ISSCC)*, vol. XIII, Feb 1970, pp. 42–43.

[11] J. C. Gealow, "Impact of Processing Technology on DRAM Sense Amplifier Design," Ph.D. dissertation, Massachusetts Institute of Technology, 1990.

[12] TechInsights Inc., "Intel 10 nm Logic Process Analysis (Cannon Lake)," *https://www.techinsights.com/blog/intel-10-nm-logic-process-analysis-cannon-lake*, 2018.

[13] S. Jones, "7nm, 5nm and 3nm Logic, Current and Projected Processes," *https://semiwiki.com/semiconductor-manufacturers/intel/7544-7nm-5nm-and-3nm-logic-current-and-projected-processes/*, Jun 2018.

[14] A. Shilov, "Samsung Completes Development of 5nm EUV Process Technology," *https://www.anandtech.com/show/14231/samsung-completes-development-of-5-nm-euv-process-technology*, Apr 2019.

[15] S. Jones, "TSMC and Samsung 5nm Comparison," *https://semiwiki.com/semiconductor-manufacturers/samsung-foundry/8157-tsmc-and-samsung-5nm-comparison/*, May 2019.

[16] D. Nenni, "Samsung vs TSMC 7nm Update," *https://semiwiki.com/semiconductor-manufacturers/samsung-foundry/7926-samsung-vs-tsmc-7nm-update/*, Jan 2019.

[17] C. Auth, A. Aliyarukunju, M. Asoro, D. Bergstrom, V. Bhagwat, J. Birdsall, N. Bisnik, M. Buehler, V. Chikarmane, G. Ding, Q. Fu, H. Gomez, W. Han, D. Hanken, M. Haran, M. Hattendorf, R. Heussner, H. Hiramatsu, B. Ho, S. Jaloviar, I. Jin, S. Joshi, S. Kirby, S. Kosaraju, H. Kothari, G. Leatherman, K. Lee, J. Leib, A. Madhavan, K. Marla, H. Meyer, T. Mule, C. Parker, S. Parthasarathy, C. Pelto, L. Pipes, I. Post,

M. Prince, A. Rahman, S. Rajamani, A. Saha, J. D. Santos, M. Sharma, V. Sharma, J. Shin, P. Sinha, P. Smith, M. Sprinkle, A. S. Amour, C. Staus, R. Suri, D. Towner, A. Tripathi, A. Tura, C. Ward, and A. Yeoh, "A 10nm High Performance and Low-power CMOS Technology Featuring 3rd generation FinFET Transistors, Self-Aligned Quad Patterning, Contact over Active Gate and Cobalt Local Interconnects," in *IEEE International Electron Devices Meeting (IEDM)*, Dec 2017, pp. 29.1.1–29.1.4.

[18] D. Hisamoto, T. Kaga, Y. Kawamoto, and E. Takeda, "A Fully Depleted Lean-channel Transistor (DELTA) - A Novel vertical Ultra Thin SOI MOSFET," in *IEEE International Electron Devices Meeting (IEDM)*. IEEE, 1989, pp. 833–836.

[19] E. Leobandung and S. Y. Chou, "Reduction of Short Channel Effects in SOI MOSFETs with 35 nm Channel Width and 70 nm Channel Length," in *54th Annual Device Research Conference Digest*, June 1996, pp. 110–111.

[20] D. Hisamoto, Wen-Chin Lee, J. Kedzierski, H. Takeuchi, K. Asano, C. Kuo, E. Anderson, Tsu-Jae King, J. Bokor, and Chenming Hu, "FinFET - A Self-aligned Double-gate MOSFET Scalable to 20 nm," *IEEE Transactions on Electron Devices*, vol. 47, no. 12, pp. 2320–2325, Dec 2000.

[21] H. Takato, K. Sunouchi, N. Okabe, A. Nitayama, K. Hieda, F. Horiguchi, and F. Masuoka, "High Performance CMOS Surrounding Gate Transistor (SGT) for Ultra High Density LSIs," in *IEEE International Electron Devices Meeting (IEDM)*, Dec 1988, pp. 222–225.

[22] H. Lee, L. . Yu, S. . Ryu, J. . Han, K. Jeon, D. . Jang, K. . Kim, J. Lee, J. . Kim, S. Jeon, G. Lee, J. Oh, Y. Park, W. Bae, H. Lee, J. Yang, J. Yoo, S. Kim, and Y. . Choi, "Sub-5nm All-Around Gate FinFET for Ultimate Scaling," in *IEEE Symposium on VLSI Technology*, June 2006, pp. 58–59.

[23] K. Mistry, C. Allen, C. Auth, B. Beattie, D. Bergstrom, M. Bost, M. Brazier, M. Buehler, A. Cappellani, R. Chau, C. . Choi, G. Ding, K. Fischer, T. Ghani, R. Grover, W. Han, D. Hanken, M. Hattendorf, J. He, J. Hicks, R. Huessner, D. Ingerly, P. Jain, R. James, L. Jong, S. Joshi, C. Kenyon, K. Kuhn, K. Lee, H. Liu, J. Maiz, B. McIntyre, P. Moon, J. Neirynck, S. Pae, C. Parker, D. Parsons, C. Prasad, L. Pipes, M. Prince, P. Ranade, T. Reynolds, J. Sandford, L. Shifren, J. Sebastian, J. Seiple, D. Simon, S. Sivakumar, P. Smith, C. Thomas, T. Troeger, P. Vandervoorn, S. Williams, and K. Zawadzki, "A 45nm Logic Technology with High-k+ Metal Gate Transistors, Strained Silicon, 9 Cu Interconnect Layers, 193nm Dry Patterning, and 100% Pb-free Packaging," in *IEEE International Electron Devices Meeting (IEDM)*, Dec 2007, pp. 247–250.

[24] H. Wu, N. Conrad, Wei Luo, and P. D. Ye, "First Experimental Demonstration of Ge CMOS Circuits," in *IEEE International Electron Devices Meeting (IEDM)*, Dec 2014, pp. 9.3.1–9.3.4.

[25] A. Toriumi, "Recent Progress of Germanium MOSFETs," in *IEEE International Meeting for Future of Electron Devices, Kansai*, May 2012, pp. 1–2.

[26] E. Capogreco, L. Witters, H. Arimura, F. Sebaai, C. Porret, A. Hikavyy, R. Loo, A. P. Milenin, G. Eneman, P. Favia, H. Bender, K. Wostyn, E. Dentoni Litta, A. Schulze, C. Vrancken, A. Opdebeeck, J. Mitard, R. Langer, F. Holsteyns, N. Waldron, K. Barla, V. De Heyn, D. Mocuta, and N. Collaert, "First Demonstration of Vertically Stacked Gate-All-Around Highly Strained Germanium Nanowire pFETs," *IEEE Transactions on Electron Devices*, vol. 65, no. 11, pp. 5145–5150, Nov 2018.

[27] K. W. Guarini, A. W. Topol, M. Ieong, R. Yu, L. Shi, M. R. Newport, D. J. Frank, D. V. Singh, G. M. Cohen, S. V. Nitta, D. C. Boyd, P. A. O'Neil, S. L. Tempest, H. B.

Pogge, S. Purushothaman, and W. E. Haensch, "Electrical Integrity of State-of-the-art 0.13 /spl mu/m SOI CMOS Devices and Circuits Transferred for Three-dimensional (3D) Integrated Circuit (IC) Fabrication," in *IEEE International Electron Devices Meeting (IEDM)*, Dec 2002, pp. 943–945.

[28] A. W. Topol, D. C. L. Tulipe, L. Shi, D. J. Frank, K. Bernstein, S. E. Steen, A. Kumar, G. U. Singco, A. M. Young, K. W. Guarini, and M. Ieong, "Three-dimensional Integrated Circuits," *IBM Journal of Research and Development*, vol. 50, no. 4.5, pp. 491–506, July 2006.

[29] K. Arabi, K. Samadi, and Y. Du, "3D VLSI: A Scalable Integration Beyond 2D," in *International Symposium on Physical Design (ISPD)*. ACM, 2015, pp. 1–7.

[30] Y. Son, B. Frost, Y. Zhao, and R. L. Peterson, "Monolithic Integration of High-voltage Thin-film Electronics on Low-voltage Integrated Circuits Using a Solution Process," *Nature Electronics*, vol. 2, no. 11, pp. 540–548, 2019.

[31] R. R. Tummala, "Moore's Law Meets its Match (System-on-Package)," *IEEE Spectrum*, vol. 43, no. 6, pp. 44–49, June 2006.

[32] N. B. Cobb and A. Zakhor, *Fast Optical and Process Proximity Correction Algorithms for Integrated Circuit Manufacturing*. Citeseer, 1998.

[33] Y. Liu and A. Zakhor, "Binary and Phase Shifting Mask Design for Optical Lithography," *IEEE Transactions on Semiconductor Manufacturing*, vol. 5, no. 2, pp. 138–152, May 1992.

[34] S. Sun, C. Bencher, Y. Chen, H. Dai, M.-P. Cai, J. Jin, P. Blanco, L. Miao, P. Xu, X. Xu, J. Yu, R. Hung, S. Oemardani, O. Chan, C.-P. Chang, and C. Ngai, "Demonstration of 32nm Half-pitch Electrical Testable NAND FLASH Patterns Using Self-Aligned Double Patterning," in *Optical Microlithography XXII*, H. J. Levinson and

M. V. Dusa, Eds., vol. 7274, International Society for Optics and Photonics. SPIE, 2009, pp. 117 – 123.

[35] Y. Ma, J. Sweis, H. Yoshida, Y. Wang, J. Kye, and H. J. Levinson, "Self-Aligned Double Patterning (SADP) Compliant Design Flow," in *Design for Manufacturability through Design-Process Integration VI*, M. E. Mason, Ed., vol. 8327, International Society for Optics and Photonics. SPIE, 2012, pp. 49 – 61.

[36] R. R. H. Kim and J. Cantone, "Methods for Fabricating Integrated Circuits Using Self-Aligned Quadruple Patterning," U.S. Patent 9,171,764, Oct 27, 2015.

[37] J. R. Cantone, L. Jang, and R. R.-H. Kim, "Methods for Fabricating Integrated Circuits Using Self-Aligned Quadruple Patterning," U.S. Patent 9,209,038, Dec, 2015.

[38] M. E. Colburn, S. K. Kanakasabapathy, F. L. Lie, and S. A. Sieg, "Self-aligned Quadruple Patterning Process," U.S. Patent 9,305,845, Apr 5, 2016.

[39] X. Dai, H. Liu, J. P. Liu, and J. Li, "Methods for Fabricating Integrated Circuits Using Self-Aligned Quadruple Patterning," U.S. Patent 9,620,380, Apr 11, 2017.

[40] D. Z. Pan, B. Yu, and J. Gao, "Design for Manufacturing With Emerging Nanolithography," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 32, no. 10, pp. 1453–1472, Oct 2013.

[41] C. W. Gwyn, R. Stulen, D. Sweeney, and D. Attwood, "Extreme Ultraviolet Lithography," *Journal of Vacuum Science & Technology B: Microelectronics and Nanometer Structures Processing, Measurement, and Phenomena*, vol. 16, no. 6, pp. 3142–3149, 1998.

[42] B. Wu and A. Kumar, "Extreme Ultraviolet Lithography: A Review," *Journal of Vacuum Science & Technology B: Microelectronics and Nanometer Structures Processing, Measurement, and Phenomena*, vol. 25, no. 6, pp. 1743–1761, 2007.

[43] J. Doweck, W. Kao, A. K. Lu, J. Mandelblat, A. Rahatekar, L. Rappoport, E. Rotem, A. Yasin, and A. Yoaz, "Inside 6th-Generation Intel Core: New Microarchitecture Code-Named Skylake," *IEEE Micro*, vol. 37, no. 2, pp. 52–62, Mar 2017.

[44] Ian Cutress, "Examining Intel's Ice Lake Processors: Taking a Bite of the Sunny Cove Microarchitecture," *https://www.anandtech.com/show/14514/examining-intels-ice-lake-microarchitecture-and-sunny-cove*, Jul 2019.

[45] T. Singh, S. Rangarajan, D. John, C. Henrion, S. Southard, H. McIntyre, A. Novak, S. Kosonocky, R. Jotwani, A. Schaefer, E. Chang, J. Bell, and M. Co, "Zen: A Next-generation High-performance x86 Core," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2017, pp. 52–53.

[46] Frumusanu, Andrei and Bonshor, Gavin, "The AMD 3rd Gen Ryzen Deep Dive Review: 3700X and 3900X Raising The Bar," *https://www.anandtech.com/show/14605/the-and-ryzen-3700x-3900x-review-raising-the-bar*, Jul 2019.

[47] R. K. Nurani, A. J. Strojwas, W. P. Maly, C. Ouyang, W. Shindo, R. Akella, M. G. McIntyre, and J. Derrett, "In-line Yield Prediction Methodologies Using Patterned Wafer Inspection Information," *IEEE Transactions on Semiconductor Manufacturing*, vol. 11, no. 1, pp. 40–47, Feb 1998.

[48] D. J. Ciplickas, Xiaolei Li, and A. J. Strojwas, "Predictive Yield Modeling of VLSIC's," in *International Workshop on Statistical Metrology*, June 2000, pp. 28–37.

[49] S. Jansen, G. Florence, A. Perry, and S. Fox, "Utilizing Design Layout Information to Improve Efficiency of SEM Defect Review Sampling," in *Advanced Semiconductor Manufacturing Conference (ASMC)*, May 2008, pp. 69–71.

[50] C. Huang, C. Young, H. Liu, S. F. Tzou, D. Tsui, A. Tsai, and E. Chang, "Using Design Based Binning to Improve Defect Excursion Control for 45nm Production," in *International Symposium on Semiconductor Manufacturing*, Oct 2007, pp. 1–3.

[51] M. G. Buehler, "Microelectronic Test Chips for VLSI Electronics," in *VLSI Electronics Microstructure Science*. Elsevier, 1983, vol. 6, pp. 529–576.

[52] H. R. Sayah and M. G. Buehler, "Comb/serpentine/cross-bridge Test Structure For Fabrication Process Evaluation," in *IEEE International Conference on Microelectronic Test Structures*, Feb 1988, pp. 23–28.

[53] C. Hess and L. H. Weiland, "Harp Test Structure to Electrically Determine Size Distributions of Killer Defects," *IEEE Transactions on Semiconductor Manufacturing*, vol. 11, no. 2, pp. 194–203, May 1998.

[54] J. Segal, A. Jee, D. Lepejian, and B. Chu, "Using Electrical Bitmap Results from Embedded Memory to Enhance Yield," *IEEE Design & Test of Computers*, vol. 18, no. 3, pp. 28–39, May 2001.

[55] R. D. Blanton, B. Niewenhuis, and C. Taylor, "Logic Characterization Vehicle Design for Maximal Information Extraction for Yield Learning," in *IEEE International Test Conference (ITC)*, Oct 2014, pp. 1–10.

[56] R. D. Blanton, B. Niewenhuis, and Z. D. Liu, "Design Reflection for Optimal Test-chip Implementation," in *IEEE International Test Conference (ITC)*, Oct 2015, pp. 1–10.

[57] Z. Liu, B. Niewenhuis, S. Mittal, and R. D. Blanton, "Achieving 100% Cell-aware Coverage by Design," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2016, pp. 109–114.

[58] B. Niewenhuis, Z. Dexter Liu, S. Mittal, and R. D. Blanton, "Logic Characterization Vehicle Design for Yyield Learning," in *Advanced Semiconductor Manufacturing Conference (ASMC)*, May 2016, pp. 375–380.

[59] S. Mittal, Z. Liu, B. Niewenhuis, and R. D. Blanton, "Test Chip Design for Optimal Cell-aware Diagnosability," in *IEEE International Test Conference (ITC)*, Nov 2016, pp. 1–8.

[60] P. Fynan, Z. Liu, B. Niewenhuis, S. Mittal, M. Strajwas, and R. D. Blanton, "Logic Characterization Vehicle Design Reflection via Layout Rewiring," in *IEEE International Test Conference (ITC)*, Nov 2016, pp. 1–10.

[61] B. Niewenhuis, S. Mittal, and R. D. Blanton, "Multiple-defect Diagnosis for Logic Characterization Vehicles," in *IEEE European Test Symposium (ETS)*, May 2017, pp. 1–6.

[62] C. Hess, B. E. Stine, L. H. Weiland, and K. Sawada, "Logic Characterization Vehicle to Determine Process Variation Impact on Yield and Performance of Digital Circuits," in *International Conference on Microelectronic Test Structures (ICMTS)*, April 2002, pp. 189–196.

[63] 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, July 1966.

[64] K. C. Y. Mei, "Bridging and Stuck-At Faults," *IEEE Transactions on Computers*, vol. C-23, no. 7, pp. 720–727, July 1974.

[65] V. Boppana and M. Fujita, "Modeling the Unknown! Towards Model-independent Fault and Error Diagnosis," in *IEEE International Test Conference (ITC)*, Oct 1998, pp. 1094–1101.

[66] G. R. Case, "Analysis of Actual Fault Mechanisms in CMOS Logic Gates," in *ACM/IEEE Design Automation Conference (DAC)*. ACM, 1976, pp. 265–270.

[67] R. L. Wadsack, "Fault Modeling and Logic Simulation of CMOS and MOS Integrated Circuits," *The Bell System Technical Journal*, vol. 57, no. 5, pp. 1449–1474, May 1978.

[68] R. D. Blanton and J. P. Hayes, "The Input Pattern Fault Model and its Application," in *Proceedings European Design and Test Conference. ED TC 97*, March 1997, pp. 628–.

[69] R. D. Blanton and J. P. Hayes, "Properties of the Input Pattern Fault Model," in *International Conference on Computer Design VLSI in Computers and Processors*, Oct 1997, pp. 372–380.

[70] R. D. Blanton and J. P. Hayes, "On the Properties of the Input Pattern Fault Model," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 8, no. 1, pp. 108–124, 2003.

[71] J. A. Acken and S. D. Millman, "Accurate Modeling and Simulation of Bridging Faults," in *IEEE Custom Integrated Circuits Conference*, May 1991.

[72] P. C. Maxwell and R. C. Aitken, "Biased Voting: A Method for Simulating CMOS Bridging Faults in the Presence of Variable Gate Logic Thresholds," in *IEEE International Test Conference (ITC)*, Oct 1993, pp. 63–72.

[73] F. Hapke, R. Krenz-Baath, A. Glowatz, J. Schloeffel, H. Hashempour, S. Eichenberger, C. Hora, and D. Adolfsson, "Defect-oriented Cell-aware ATPG and Fault Simulation for

Industrial Cell Libraries and Designs," in *IEEE International Test Conference (ITC)*, Nov 2009, pp. 1–10.

[74] 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 (TCAD)*, vol. 33, no. 9, pp. 1396–1409, Sept 2014.

[75] F. Hapke and J. Schloeffel, "Introduction to the Defect-oriented Cell-aware Test Methodology for Significant Reduction of DPPM rates," in *IEEE European Test Symposium (ETS)*, May 2012, pp. 1–6.

[76] H. Liu, B. Lin, and C. Wu, "Layout-Oriented Defect Set Reduction for Fast Circuit Simulation in Cell-Aware Test," in *IEEE Asian Test Symposium (ATS)*, Nov 2016, pp. 156–160.

[77] F. Hapke, W. Redemund, J. Schloeffel, R. Krenz-Baath, A. Glowatz, M. Wittke, H. Hashempour, and S. Eichenberger, "Defect-oriented Cell-internal Testing," in *IEEE International Test Conference (ITC)*, Nov 2010, pp. 1–10.

[78] F. Hapke, J. Schloeffel, W. Redemund, A. Glowatz, J. Rajski, M. Reese, J. Rearick, and J. Rivers, "Cell-aware Analysis for Small-delay Effects and Production Test Results from Different Fault Models," in *IEEE International Test Conference (ITC)*, Sep. 2011, pp. 1–8.

[79] F. J. Ferguson and J. P. Shen, "Extraction and Simulation of Realistic CMOS Faults Using Inductive Fault Analysis," in *IEEE International Test Conference (ITC)*, Sep 1988, pp. 475–484.

[80] J. P. Shen, W. Maly, and F. J. Ferguson, "Inductive Fault Analysis of MOS Integrated Circuits," *IEEE Design & Test of Computers*, vol. 2, no. 6, pp. 13–26, Dec 1985.

[81] G. L. Smith, "Model for Delay Faults Based Upon Paths," in *IEEE International Test Conference (ITC)*. Citeseer, 1985, pp. 342–351.

[82] T. M. Storey and J. W. Barry, "Delay Test Simulation," in *ACM/IEEE Design Automation Conference (DAC)*. IEEE Press, 1977, pp. 492–494.

[83] J. A. Waicukauski, E. Lindbloom, B. K. Rosen, and V. S. Iyengar, "Transition Fault Simulation," *IEEE Design & Test of Computers*, vol. 4, no. 2, pp. 32–38, April 1987.

[84] S. C. Ma, P. Franco, and E. J. McCluskey, "An Experimental Chip to Evaluate Test Techniques Experiment Results," in *IEEE International Test Conference (ITC)*, Oct 1995, pp. 663–672.

[85] I. Pomeranz and S. M. Reddy, "Stuck-at Tuple-detection: A Fault Model Based on Stuck-at Faults for Improved Defect Coverage," in *IEEE VLSI Test Symposium (VTS)*, April 1998, pp. 289–294.

[86] E. J. McCluskey, "Quality and Single-stuck Faults," in *IEEE International Test Conference (ITC)*, Oct 1993, pp. 597–.

[87] Kyoung Youn Cho, S. Mitra, and E. J. McCluskey, "Gate Exhaustive Testing," in *IEEE International Test Conference (ITC)*, Nov 2005, pp. 7 pp.–777.

[88] H. Y. Chang, E. Manning, G. A. Metze *et al.*, *Fault Diagnosis of Digital Systems*. Wiley-Interscience, 1970.

[89] 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, July 1966.

[90] J. Richman and K. R. Bowden, "The Modern Fault Dictionary," in *IEEE International Test Conference (ITC)*, Sept 1985, pp. 696–702.

[91] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design*. Computer science press New York, 1990, vol. 2.

[92] I. Pomeranz and S. M. Reddy, "On the Generation of Small Dictionaries for Fault Location," in *IEEE International Conference on Computer-Aided Design (ICCAD)*, Nov 1992, pp. 272–279.

[93] P. G. Ryan, W. K. Fuchs, and I. Pomeranz, "Fault Dictionary Compression and Equivalence Class Computation for Sequential Circuits," in *IEEE International Conference on Computer Aided Design (ICCAD)*, Nov 1993, pp. 508–511.

[94] V. Boppana, I. Hartanto, and W. K. Fuchs, "Full Fault Dictionary Storage based on Labeled Tree Encoding," in *IEEE VLSI Test Symposium (VTS)*, Apr 1996, pp. 174–179.

[95] B. Chess and T. Larrabee, "Creating Small Fault Dictionaries [Logic Circuit Fault Diagnosis]," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 18, no. 3, pp. 346–356, Mar 1999.

[96] D. B. Lavo and T. Larrabee, "Making Cause-Effect Cost Effective: Low-Resolution Fault Dictionaries," in *IEEE International Test Conference (ITC)*, 2001, pp. 278–286.

[97] V. Boppana and W. K. Fuchs, "Fault Dictionary Compaction by Output Sequence Removal," in *IEEE International Conference on Computer-Aided Design (ICCAD)*. IEEE Computer Society Press, 1994, pp. 576–579.

[98] P. G. Ryan, W. K. Fuchs, and I. Pomeranz, "Fault Dictionary Compression and Equivalence Class Computation for Sequential Circuits," in *IEEE International Conference on Computer Aided Design (ICCAD)*, Nov 1993, pp. 508–511.

[99] I. Pomeranz and S. M. Reddy, "A Same/Different Fault Dictionary: An Extended Pass/Fail Fault Dictionary with Improved Diagnostic Resolution," in *Design, Automation and Test in Europe (DATE)*, March 2008, pp. 1474–1479.

[100] R. Tulloss, "Size Optimization of Fault Dictionaries," in *Proc. Semiconductor Test Conf*, 1978, pp. 264–265.

[101] R. Tulloss, "Fault Dictionary Compression: Recognizing When a Fault may be Unambiguously Represented by a Single Failure Detection," in *IEEE International Test Conference (ITC)*, 1980, pp. 368–370.

[102] I. Pomeranz, "On Pass/fail Dictionaries for Scan Circuits," in *IEEE Asian Test Symposium (ATS)*, Nov 2001, pp. 51–56.

[103] B. Arslan and A. Orailoglu, "Fault Dictionary Size Reduction Through Test Response Superposition," in *IEEE International Conference on Computer Design: VLSI in Computers and Processors*, Sep. 2002, pp. 480–485.

[104] M. Abramovici and M. A. Breuer, "Multiple Fault Diagnosis in Combinational Circuits Based on an Effect-cause Analysis," *IEEE Transactions on Computers*, vol. C-29, no. 6, pp. 451–460, June 1980.

[105] H. Cox and J. Rajski, "A Method of Fault Analysis for Test Generation and Fault Diagnosis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 7, no. 7, pp. 813–833, Jul 1988.

[106] J. Savir and J. Roth, "Testing for, and Distinguishing Between Failures," in *Proceedings of the Fault Tolerant Computing Symposium*, June 1982, pp. 165–172.

[107] T. Ramakrishnan and L. Kinney, "Extension of the Critical Path Tracing Algorithm," in *ACM/IEEE Design Automation Conference (DAC)*, June 1990, pp. 720–723.

[108] S. Venkataraman and S. B. Drummonds, "A Technique for Logic Fault Diagnosis of Interconnect Open Defects," in *IEEE VLSI Test Symposium (VTS)*, April 2000, pp. 313–318.

[109] Venkataraman and Fuchs, "A Deductive Technique for Diagnosis of Bridging Faults," in *IEEE International Conference on Computer Aided Design (ICCAD)*, Nov 1997, pp. 562–567.

[110] S. Venkataraman, I. Hartanto, and W. Kent Fuchs, "Dynamic Diagnosis of Sequential Circuits Based on Stuck-at Faults," in *IEEE VLSI Test Symposium (VTS)*, April 1996, pp. 198–203.

[111] M. Abramovici, P. R. Menon, and D. T. Miller, "Critical Path Tracing - An Alternative to Fault Simulation," in *IEEE Design Automation Conference (DAC)*. IEEE, 1983, pp. 214–220.

[112] P. R. Menon, Y. Levendel, and M. Abramovici, "Critical Path tracing in Sequential Circuits," in *IEEE International Conference on Computer-Aided Design (ICCAD)*, Nov 1988, pp. 162–165.

[113] P. R. Menon, Y. H. Levendel, and M. Abramovici, "SCRIPT: A Critical Path Tracing Algorithm for Synchronous Sequential Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 10, pp. 738–747, 1991.

[114] S. B. Akers, S. Park, B. Krishnamurthy, and A. Swaminathan, "Why is Less Information from Logic Simulation More Useful in Fault Simulation?" in *IEEE International Test Conference (ITC)*, Sep. 1990, pp. 786–800.

[115] S. Venkataraman and S. B. Drummonds, "POIROT: a logic fault diagnosis tool and its applications," in *IEEE International Test Conference (ITC)*, Oct 2000, pp. 253–262.

[116] W. Zou, W. Cheng, S. M. Reddy, and H. Tang, "Speeding Up Effect-Cause Defect Diagnosis Using a Small Dictionary," in *IEEE VLSI Test Symposium (VTS)*, May 2007, pp. 225–230.

[117] H. Tang, C. Liu, W. Cheng, S. M. Reddy, and W. Zou, "Improving Performance of Effect-Cause Diagnosis with Minimal Memory Overhead," in *IEEE Asian Test Symposium (ATS)*, Oct 2007, pp. 281–287.

[118] P. G. Ryan, S. Rawat, and W. K. Fuchs, "Two-stage Fault Location," in *IEEE International Test Conference (ITC)*. IEEE, 1991, p. 963.

[119] J. C. M. Li and E. J. McCluskey, "Diagnosis of Sequence-dependent Chips," in *IEEE VLSI Test Symposium (VTS)*, 2002, pp. 187–192.

[120] B. Bosio, P. Girard, S. Pravossoudovitch, and A. Virazel, "A Comprehensive Framework for Logic Diagnosis of Arbitrary Defects," *IEEE Transactions on Computers*, vol. 59, no. 3, pp. 289–300, March 2010.

[121] Y. Benabboud, A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, L. Bouzaida, and I. Izaute, "Comprehensive Bridging Fault Diagnosis Based on the SLAT Paradigm," in *International Symposium on Design and Diagnostics of Electronic Circuits Systems*, April 2009, pp. 264–269.

[122] Y. Benabboud, A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, L. Bouzaida, and I. Izaute, "A Fault-simulation-based Approach for Logic Diagnosis," in *International Conference on Design Technology of Integrated Systems in Nanoscale Era*, April 2009, pp. 216–222.

[123] A. Rousset, A. Bosio, P. Girard, C. Landrault, S. Pravossoudovitch, and A. Virazel, "Fast Bridging Fault Diagnosis using Logic Information," in *IEEE Asian Test Symposium (ATS)*, Oct 2007, pp. 33–38.

[124] S. D. Millman, E. J. McCluskey, and J. M. Acken, "Diagnosing CMOS Bridging Faults with Stuck-at Fault Dictionaries," in *IEEE International Test Conference (ITC)*, Sep. 1990, pp. 860–870.

[125] B. Chess, D. B. Lavo, F. J. Ferguson, and T. Larrabee, "Diagnosis of Realistic Bridging Faults with Single Stuck-at Information," in *IEEE International Conference on Computer Aided Design (ICCAD)*, Nov 1995, pp. 185–192.

[126] D. B. Lavo, B. Chess, T. Larrabee, and F. J. Ferguson, "Diagnosing Realistic Bridging Faults with Single Stuck-at Information," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 17, no. 3, pp. 255–268, March 1998.

[127] D. B. Lavo, T. Larrabee, and B. Chess, "Beyond the Byzantine Generals: Unexpected Behavior and Bridging Fault Diagnosis," in *IEEE International Test Conference (ITC)*, Oct 1996, pp. 611–619.

[128] S. Drummonds *et al.*, "Bridging the Gap Between Logical Diagnosis and Physical Analysis," in *IEEE International Workshop on Defect Based Testing*, 2002, pp. 272–279.

[129] J. Saxena, K. M. Butler, H. Balachandran, D. B. Lavo, B. Chess, T. Larrabee, and F. J. Ferguson, "On Applying Non-classical Defect Models to Automated Diagnosis," in *IEEE International Test Conference (ITC)*, Oct 1998, pp. 748–757.

[130] Chakravarty and Y. Gong, "Voting Model Based Diagnosis of Bridging Faults in Combinational Circuits," in *IEEE International Conference on VLSI Design (VLSID)*, Jan 1995, pp. 338–342.

[131] J. A. Waicukauski and E. Lindbloom, "Failure Diagnosis of Structured VLSI," *IEEE Design & Test of Computers*, vol. 6, no. 4, pp. 49–60, Aug 1989.

[132] I. Yamazaki, H. Yamanaka, T. Ikeda, M. Takakura, and Y. Sato, "An Approach to Improve the Resolution of Defect-based Diagnosis," in *IEEE Asian Test Symposium (ATS)*, Nov 2001, pp. 123–128.

[133] C. Liu, "Improve the Quality of Per-Test Fault Diagnosis Using Output Information," *Journal of Electronic Testing*, vol. 23, no. 1, pp. 11–24, Feb 2007.

[134] Chunsheng Liu, "An Efficient Method for Improving the Quality of Per-test Fault Diagnosis," in *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, Nov 2004, pp. 648–651.

[135] Lixing Zhao and V. D. Agrawal, "Net Diagnosis Using Stuck-at and Transition Fault Models," in *IEEE VLSI Test Symposium (VTS)*, April 2012, pp. 221–226.

[136] T. Bartenstein, D. Heaberlin, L. Huisman, and D. Sliwinski, "Diagnosing Combinational Logic Designs Using the Single Location at-a-time (SLAT) Paradigm," in *IEEE International Test Conference (ITC)*, 2001, pp. 287–296.

[137] L. M. Huisman, "Diagnosing Arbitrary Defects in Logic Designs Using Single Location at a Time (SLAT)," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 23, no. 1, pp. 91–101, 2004.

[138] S. Holst and H. Wunderlich, "Adaptive Debug and Diagnosis Without Fault Dictionaries," in *IEEE European Test Symposium (ETS)*, May 2007, pp. 7–12.

[139] S. Holst and H.-J. Wunderlich, "Adaptive Debug and Diagnosis Without Fault Dictionaries," *Journal of Electronic Testing*, vol. 25, no. 4-5, pp. 259–268, 2009.

[140] Wei Zou, Wu-Tung Cheng, S. M. Reddy, and Huaxing Tang, "On Methods to Improve Location Based Logic Diagnosis," in *International Conference on VLSI Design held*

jointly with 5th International Conference on Embedded Systems Design (VLSID), Jan 2006, pp. 7 pp.–.

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

[142] R. Desineni and R. D. Blanton, "Diagnosis of Arbitrary Defects Using Neighborhood Function Extraction," in *IEEE VLSI Test Symposium (VTS)*, May 2005, pp. 366–373.

[143] R. Desineni, O. Poku, and R. D. Blanton, "A Logic Diagnosis Methodology for Improved Localization and Extraction of Accurate Defect Behavior," in *IEEE International Test Conference (ITC)*, Oct 2006, pp. 1–10.

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

[145] Y. Sato, L. Yamazaki, H. Yamanaka, T. Ikeda, and M. Takakura, "A Persistent Diagnostic Technique for Unstable Defects," in *IEEE International Test Conference (ITC)*, 2002, pp. 242–249.

[146] S. Mittal and R. D. Blanton, "PADLOC: Physically-aware Defect Localization and Characterization," in *IEEE Asian Test Symposium (ATS)*, Nov 2017.

[147] S.-Y. Huang, "Diagnosis of Byzantine Open-segment Faults [Scan Testing]," in *IEEE Asian Test Symposium (ATS)*, Nov 2002, pp. 248–253.

[148] W. Zou, W. T. Cheng, and S. M. Reddy, "Interconnect Open Defect Diagnosis with Physical Information," in *IEEE Asian Test Symposium (ATS)*, Nov 2006, pp. 203–209.

[149] D. Arumi, R. Rodriguez-Montanes, J. Figueras, S. Eichenberger, C. Hora, and B. Kruseman, "Diagnosis of Interconnect Full Open Defects in the Presence of Fan-Out," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 30, no. 12, pp. 1911–1922, Dec 2011.

[150] R. Rodriguez-Montanes, D. Arumi, J. Figueras, S. Einchenberger, C. Hora, B. Kruseman, M. Lousberg, and A. K. Majhi, "Diagnosis of Full Open Defects in Interconnecting Lines," in *IEEE VLSI Test Symposium (VTS)*, May 2007, pp. 158–166.

[151] D. Arumi, R. Rodriguez-Montanes, J. Figueras, S. Eichenberger, C. Hora, and B. Kruseman, "Diagnosis of Interconnect Full Open Defects in the Presence of Fan-Out," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 30, no. 12, pp. 1911–1922, Dec 2011.

[152] D. Arumi, R. Rodriguez-Montanes, J. Figueras, S. Eichenberger, C. Hora, and B. Kruseman, "Diagnosis of Interconnect Full Open Defects in the Presence of Gate Leakage Currents," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 32, no. 2, pp. 301–312, Feb 2013.

[153] C. Liu, W. Zou, S. M. Reddy, W.-T. Cheng, M. Sharma, and H. Tang, "Interconnect Open Defect Diagnosis with Minimal Physical Information," in *IEEE International Test Conference (ITC)*, Oct 2007, pp. 1–10.

[154] K. Yamazaki, T. Tsutsumi, H. Takahashi, Y. Higami, T. Aikyo, Y. Takamatsu, H. Yotsuyanagi, and M. Hashizume, "A Novel Approach for Improving the Quality of Open Fault Diagnosis," in *IEEE International Conference on VLSI Design (VLSID)*, Jan 2009, pp. 85–90.

[155] H. Takahashi, Y. Higami, S. Kadoyama, T. Aikyo, Y. Takamatsu, K. Yamazaki, T. Tsutsumi, H. Yotsuyanagi, and M. Hashizume, "Clues for Modeling and Diagnos-

ing Open Faults with Considering Adjacent Lines," in *IEEE Asian Test Symposium (ATS)*, Oct 2007, pp. 39–44.

[156] Chakravarty and Y. Gong, "Voting Model Based Diagnosis of Bridging Faults in Combinational Circuits," in *IEEE International Conference on VLSI Design (VLSID)*, Jan 1995, pp. 338–342.

[157] W. Zou, W.-T. Cheng, and S. M. Reddy, "Bridge Defect Diagnosis with Physical Information," in *IEEE Asian Test Symposium (ATS)*, Dec 2005, pp. 248–253.

[158] M. Keim, N. Tamarapalli, H. Tang, M. Sharma, J. Rajski, C. Schuermyer, and B. Benware, "A Rapid Yield Learning Flow Based on Production Integrated Layout-Aware Diagnosis," in *IEEE International Test Conference (ITC)*, Oct 2006, pp. 1–10.

[159] M. Sharma, B. Benware, L. Ling, D. Abercrombie, L. Lee, M. Keim, H. Tang, W. T. Cheng, T. P. Tai, Y. J. Chang, R. Lin, and A. Man, "Efficiently Performing Yield Enhancements by Identifying Dominant Physical Root Cause from Test Fail Data," in *IEEE International Test Conference (ITC)*, Oct 2008, pp. 1–9.

[160] Y.-J. Chang, M.-T. Pang, M. Brennan, A. Man, M. Keim, G. Eide, B. Benware, and T.-P. Tai, "Experiences with Layout-aware Diagnosis - A Case Study," in *Electronic Device Failure Analysis*, vol. 12, 2010, pp. 12–18.

[161] M. Sharma, S. Schwarz, J. Schmerberg, K. Yang, T.-P. Tai, Y.-S. Chen, C.-Y. Chuang, and F.-M. Kuo, "Layout-aware Diagnosis Leads to Efficient and Effective Physical Failure Analysis," in *International Symposium for Testing and Failure Analysis (ISTFA)*, 2011, pp. 87–90.

[162] S. Mittal and R. D. Blanton, "NOIDA: Noise-resistant Intra-cell Diagnosis," in *IEEE VLSI Test Symposium (VTS)*, April 2018.

[163] Xinyue Fan, W. Moore, C. Hora, and G. Gronthoud, "A Novel Stuck-at Based Method for Transistor Stuck-open Fault Diagnosis," in *IEEE International Test Conference (ITC)*, Nov 2005.

[164] Xinyue Fan, W. Moore, C. Hora, M. Konijnenburg, and G. Gronthoud, "A Gate-level Method for Transistor-level Bridging fault Diagnosis," in *IEEE VLSI Test Symposium (VTS)*, April 2006, pp. 6 pp.–271.

[165] X. Fan, W. R. Moore, C. Hora, and G. Gronthoud, "Extending Gate-level Diagnosis Tools to CMOS Intra-gate Faults," *IET Computers Digital Techniques*, vol. 1, no. 6, pp. 685–693, Nov 2007.

[166] J. C. M. Li, Chao-Wen Tseng, and E. J. McCluskey, "Testing for Resistive Opens and Stuck Opens," in *IEEE International Test Conference (ITC)*, Nov 2001, pp. 1049–1058.

[167] J. C. M. Li and E. J. McCluskey, "Diagnosis of Sequence-dependent Chips," in *IEEE VLSI Test Symposium (VTS)*, 2002, pp. 187–192.

[168] Z. Sun, A. Bosio, L. Dilillo, P. Girard, A. Todri, A. Virazel, and E. Auvray, "Effect-cause Intra-cell Diagnosis at Transistor Level," in *International Symposium on Quality Electronic Design (ISQED)*, March 2013, pp. 460–467.

[169] Z. Sun, A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovich, A. Virazel, and E. Auvray, "Intra-cell Defects Diagnosis," *Journal of Electronic Testing*, vol. 30, no. 5, pp. 541–555, 2014.

[170] C. W. Tzeng, H. C. Cheng, and S. Y. Huang, "Layout-based Defect-driven Diagnosis for Intracell Bridging Defects," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 28, no. 5, pp. 764–769, May 2009.

[171] M. E. Amyeen, D. Nayak, and S. Venkataraman, "Improving Precision Using Mixed-level Fault Diagnosis," in *IEEE International Test Conference (ITC)*, Oct 2006, pp. 1–10.

[172] A. Ladhar, M. Masmoudi, and L. Bouzaida, "Efficient and Accurate Method for Intra-gate Defect Diagnoses in Nanometer Technology and Volume Data," in *Design, Automation Test in Europe Conference (DATE)*, Apr 2009, pp. 988–993.

[173] A. Ladhar, Laroussi Bouzaida, and Mohamed Masmoudi, "Layout Based Method to Diagnose Intra-gate Defects in Presence of Multiple-fault," in *International Conference on Signals, Circuits and Systems*, Nov 2008, pp. 1–6.

[174] A. Ladhar and M. Masmoudi, "An Effective and Accurate Methodology for the Cell Internal Defect Diagnosis," *Journal of Electronic Testing*, vol. 26, no. 6, pp. 621–639, 2010.

[175] F. Hapke, M. Reese, J. Rivers, A. Over, V. Ravikumar, W. Redemund, A. Glowatz, J. Schloeffel, and J. Rajski, "Cell-aware Production Test Results from a 32-nm Notebook Processor," in *IEEE International Test Conference (ITC)*, Nov 2012, pp. 1–9.

[176] H. Tang, B. Benware, M. Reese, J. Caroselli, T. Herrmann, F. Hapke, R. Tao, W. T. Cheng, and M. Sharma, "Diagnosing Cell Internal Defects Using Analog Simulation-based Fault Models," in *IEEE Asian Test Symposium (ATS)*, Nov 2014, pp. 318–323.

[177] P. Maxwell, F. Hapke, and H. Tang, "Cell-aware Diagnosis: Defective Inmates Exposed in their Cells," in *IEEE European Test Symposium (ETS)*, May 2016, pp. 1–6.

[178] F. Hapke, M. Keim, T. Herrmann, M. Reese *et al.*, "Improving Failure Analysis for Cell-internal Defects through Cell Aware Technology," in *International Symposium for Testing and Failure Analysis (ISTFA)*, 2013.

[179] H. Tang, T. P. Tai, W. T. Cheng, B. Benware, and F. Hapke, "Diagnosing Timing Related Cell Internal Defects for FinFET Technology," in *VLSI Design, Automation and Test*, Apr 2015, pp. 1–4.

[180] A. Sinha, S. Pandey, A. Singhal, A. Sanyal, and A. Schmaltz, "DFM-aware Fault Model and ATPG for Intra-cell and Inter-cell Defects," in *IEEE International Test Conference (ITC)*, Oct 2017, pp. 1–10.

[181] J. Khare, W. Maly, and N. Tiday, "Fault Characterization of Standard Cell Libraries Using Inductive Contamination Analysis (ICA)," in *IEEE VLSI Test Symposium (VTS)*, April 1996, pp. 405–413.

[182] I. Pomeranz, "New Targets for Diagnostic Test Generation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (ICCAD)*, pp. 1–1, 2019.

[183] N. Wang, I. Pomeranz, B. Benware, M. EnamulAmyeen, and S. Venkataraman, "Improving the Resolution of Multiple Defect Diagnosis by Removing and Selecting Tests," in *IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Oct 2018, pp. 1–6.

[184] I. Pomeranz, "Diagnostic Test Generation That Addresses Diagnostic Holes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (ICCAD)*, vol. 38, no. 2, pp. 335–344, Feb 2019.

[185] I. Pomeranz, M. E. Amyeen, and S. Venkataraman, "Test Modification for Reduced Volumes of Fail Data," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 22, no. 4, p. 67, 2017.

[186] S. Venkataraman, I. Pomeranz, S. Bodhe, and M. E. Amyeen, "Test Reordering for Improved Scan Chain Diagnosis Using an Enhanced Defect Diagnosis Procedure," in *IEEE International Test Conference (ITC)*, Oct 2017, pp. 1–9.

[187] I. Pomeranz, "A Test Selection Procedure for Improving the Accuracy of Defect Diagnosis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 8, pp. 2759–2767, Aug 2016.

[188] I. Pomeranz, "Improving the Accuracy of Defect Diagnosis by Adding and Removing Tests," *IET Computers & Digital Techniques*, vol. 10, no. 2, pp. 47–53, 2016.

[189] I. Pomeranz, "Improving the Accuracy of Defect Diagnosis with Multiple Sets of Candidate Faults," *IEEE Transactions on Computers*, vol. 65, no. 7, pp. 2332–2338, July 2016.

[190] I. Pomeranz, "Improving the Accuracy of Defect Diagnosis by Considering Fewer Tests," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 33, no. 12, pp. 2010–2014, Dec 2014.

[191] I. Pomeranz and S. M. Reddy, "Output-Dependent Diagnostic Test Generation," in *IEEE International Conference on VLSI Design (VLSID)*, Jan 2010, pp. 3–8.

[192] G. Chen, J. Rajski, S. Reddy, and I. Pomeranz, "N-distinguishing Tests for Enhanced Defect Diagnosis," in *IEEE Asian Test Symposium (ATS)*, Nov 2009, pp. 183–186.

[193] I. Pomeranz and S. M. Reddy, "Diagnostic Test Generation Based on Subsets of Faults," in *IEEE European Test Symposium (ETS)*, May 2007, pp. 151–158.

[194] A. Veneris, R. Chang, M. S. Abadir, and M. Amiri, "Fault Equivalence and Diagnostic Test Generation using ATPG," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 5, May 2004, pp. V–V.

[195] P. Girard, C. Landrault, S. Pravossoudovitch, and B. Rodriguez, "A Diagnostic ATPG for Delay Faults Based on Genetic Algorithms," in *IEEE International Test Conference (ITC)*, Oct 1996, pp. 286–293.

[196] Huaxing Tang, Gang Chen, S. M. Reddy, Chen Wang, J. Rajski, and I. Pomeranz, "Defect Aware Test Patterns," in *Design, Automation and Test in Europe Conference (DATE)*, March 2005, pp. 450–455 Vol. 1.

[197] Y. Zhang and V. D. Agrawal, "A Diagnostic Test Generation System," in *IEEE International Test Conference (ITC)*, Nov 2010, pp. 1–9.

[198] N. K. Bhatti and R. D. Blanton, "Diagnostic Test Generation for Arbitrary Faults," in *IEEE International Test Conference (ITC)*, Oct 2006, pp. 1–9.

[199] P. Camurati, D. Medina, P. Prinetto, and M. Sonza Reorda, "A Diagnostic Test Pattern Generation Algorithm," in *IEEE International Test Conference (ITC)*, Sep. 1990, pp. 52–58.

[200] P. Camurati, A. Lioy, P. Prinetto, and M. Sonza Reorda, "Diagnosis Oriented Test Pattern Generation," in *IEEE Design Automation Conference (DAC)*, March 1990, pp. 470–474.

[201] T. Gruning, U. Mahlstedt, and H. Koopmeiners, "DIATEST: A Fast Diagnostic Test Pattern Generator for Combinational Circuits," in *IEEE International Conference on Computer-Aided Design (ICCAD)*, Nov 1991, pp. 194–197.

[202] T. Bartenstein, "Fault Distinguishing Pattern Generation," in *IEEE International Test Conference (ITC)*, Oct 2000, pp. 820–828.

[203] J. Ye, X. Zhang, Y. Hu, and X. Li, "Substantial Fault Pair At-a-Time (SFPAT): An Automatic Diagnostic Pattern Generation Method," in *IEEE Asian Test Symposium (ATS)*, Dec 2010, pp. 192–197.

[204] M. E. Amyeen, D. Kim, M. Chandrasekar, M. Noman, S. Venkataraman, A. Jain, N. Goel, and R. Sharma, "A Novel Diagnostic Test Generation Methodology and its

Application in Production Failure Isolation," in *IEEE International Test Conference (ITC)*, Nov 2016, pp. 1–10.

[205] M. Pradhan and B. B. Bhattacharya, "COMEDI: Combinatorial Election of Diagnostic Vectors From Detection Test Sets for Logic Circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 4, pp. 1467–1476, April 2017.

[206] S. Kundu, S. Chattopadhyay, I. Sengupta, and R. Kapur, "A Diagnosability Metric for Test Set Selection Targeting Better Fault Detection," in *IEEE International Conference on VLSI Design (VLSID)*, Jan 2012, pp. 436–441.

[207] M. Chandrasekar, N. P. Rahagude, and M. S. Hsiao, "Search State Compatibility Based Incremental Learning Framework and Output Deviation Based X-filling for Diagnostic Test Generation," *Journal of Electronic Testing*, vol. 26, no. 2, pp. 165–176, 2010.

[208] Xiaoming Yu, Jue Wu, and E. M. Rudnick, "Diagnostic Test Generation for Sequential Circuits," in *IEEE International Test Conference (ITC)*, Oct 2000, pp. 225–234.

[209] F. Corno, P. Prinetto, M. Rebaudengo, and M. Sonza Reorda, "GARDA: A Diagnostic ATPG for Large Synchronous Sequential Circuits," in *Proceedings the European Design and Test Conference. ED TC*, March 1995, pp. 267–271.

[210] I. Pomeranz, "Gradual Diagnostic Test Generation and Observation Point Insertion Based on the Structural Distance Between Indistinguished Fault Pairs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 6, pp. 1026–1035, June 2012.

[211] I. Pomeranz and S. M. Reddy, "A Diagnostic Test Generation Procedure for Synchronous Sequential Circuits Based on Test Elimination," in *IEEE International Test Conference (ITC)*, Oct 1998, pp. 1074–1083.

[212] I. Pomeranz, "Gradual Diagnostic Test Generation and Observation Point Insertion Based on the Structural Distance Between Indistinguished Fault Pairs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 6, pp. 1026–1035, June 2012.

[213] S. Y. H. Su and H. Ma, "Designs for Diagnosability and Reliability in VLSI systems," in *IEEE International Test Conference (ITC)*, Sep. 1988, pp. 888–897.

[214] I. Pomeranz, S. Venkataraman, and S. M. Reddy, "Z-DFD: Design-for-diagnosability Based on the Concept of Z-detection," in *IEEE International Test Conference (ITC)*, Oct 2004, pp. 489–497.

[215] F. Wang, Y. Hu, H. Li, and X. Li, "A Design-for-diagnosis Technique for Diagnosing Both Scan Chain Faults and Combinational Circuit Faults," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE Computer Society Press, 2008, pp. 571–576.

[216] Z. Li, S. K. Goel, F. Lee, and K. Chakrabarty, "Efficient Observation-point Insertion for Diagnosability Enhancement in Digital Circuits," in *IEEE International Test Conference (ITC)*, Oct 2015, pp. 1–10.

[217] C. Schuermyer, K. Cota, R. Madge, and B. Benware, "Identification of Systematic Yield Limiters in Complex ASICs through Volume Structural Test Fail Data Visualization and Analysis," in *IEEE International Test Conference (ITC)*, Nov 2005, pp. 9 pp.–145.

[218] R. Desineni, L. Pastel, M. Kassab, M. F. Fayaz, and J. Lee, "Identifying Design Systematics using Learning based Diagnostic Analysis," in *Advanced Semiconductor Manufacturing Conference (ASMC)*, July 2010, pp. 317–321.

[219] R. Kapur, M. Kapur, and M. Kapur, "Systemic Diagnostics for Increasing Wafer Yield," U.S. Patent 8,660,818, Feb 25, 2014.

[220] S. Eichenberger, J. Geuzebroek, C. Hora, B. Kruseman, and A. Majhi, "Towards a World Without Test Escapes: The Use of Volume Diagnosis to Improve Test Quality," in *IEEE International Test Conference (ITC)*, Oct 2008, pp. 1–10.

[221] W. Cheng, R. Klingenberg, B. Benware, W. Yang, M. Sharma, G. Eide, Y. Tian, S. M. Reddy, Y. Pan, S. Fernandes, and A. Chittora, "Automatic Identification of Yield Limiting Layout Patterns Using Root Cause Deconvolution on Volume Scan Diagnosis Data," in *IEEE Asian Test Symposium (ATS)*, Nov 2017, pp. 219–224.

[222] H. Tang, S. Manish, J. Rajski, M. Keim, and B. Benware, "Analyzing Volume Diagnosis Results with Statistical Learning for Yield Improvement," in *IEEE European Test Symposium (ETS)*, May 2007, pp. 145–150.

[223] W. Cheng, Yue Tian, and S. M. Reddy, "Volume Diagnosis Data Mining," in *IEEE European Test Symposium (ETS)*, May 2017, pp. 1–10.

[224] C. Shan, P. Babighian, Y. Pan, J. Carulli, and L. Wang, "Systematic Defect Detection Methodology for Volume Diagnosis: A Data Mining Perspective," in *IEEE International Test Conference (ITC)*, Oct 2017, pp. 1–10.

[225] R. Turakhia, M. Ward, S. K. Goel, and B. Benware, "Bridging DFM Analysis and Volume Diagnostics for Yield Learning - A Case Study," in *IEEE VLSI Test Symposium (VTS)*, May 2009, pp. 167–172.

[226] Y. Tian, G. Veda, W. Cheng, M. Sharma, H. Tang, N. Bawaskar, and S. M. Reddy, "A Supervised Machine Learning Application in Volume Diagnosis," in *IEEE European Test Symposium (ETS)*, May 2019, pp. 1–6.

[227] M. Sharma, C. Schuermyer, and B. Benware, "Determination of Dominant-Yield-Loss Mechanism with Volume Diagnosis," *IEEE Design & Test of Computers*, vol. 27, no. 3, pp. 54–61, May 2010.

[228] Y. Pan, A. Chittora, K. Sekar, G. S. Huat, Y. G. Feng, A. Viswanatha, and J. Lam, "Leveraging Root Cause Deconvolution Analysis for Logic Yield Ramping," in *International Symposium for Testing and Failure Analysis (ISTFA)*, 2013, pp. 602–607.

[229] C. Schuermyer, S. Malik, and T. Herrmann, "Identifying Systematic Critical Features using Silicon Diagnosis Data," in *Advanced Semiconductor Manufacturing Conference (ASMC)*, May 2012, pp. 1–6.

[230] H. Tang, M. Sharma, W. Cheng, G. Veda, D. Gehringer, M. Knowles, J. D'Souza, K. Sekar, N. Bawaskar, and Y. Pan, "Yield Learning for Complex FinFET Defect Mechanisms Based on Volume Scan Diagnosis Results," in *Advanced Semiconductor Manufacturing Conference (ASMC)*, May 2019, pp. 1–7.

[231] W. C. Tam, O. Poku, and R. D. Blanton, "Systematic defect identification through layout snippet clustering," in *IEEE International Test Conference (ITC)*, Nov 2010, pp. 1–10.

[232] W. C. Tam and R. D. Blanton, "LASIC: Layout Analysis for Systematic IC-Defect Identification Using Clustering," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (ICCAD)*, vol. 34, no. 8, pp. 1278–1290, Aug 2015.

[233] X. Yu and R. D. Blanton, "Estimating Defect-Type Distributions through Volume Diagnosis and Defect Behavior Attribution," in *IEEE International Test Conference (ITC)*, Nov 2010, pp. 1–10.

[234] X. Yu, Y. Lin, W. Tam, O. Poku, and R. D. Blanton, "Controlling DPPM through Volume Diagnosis," in *IEEE VLSI Test Symposium (VTS)*, May 2009, pp. 134–139.

[235] X. Yu and R. D. Blanton, "Diagnosis-Assisted Adaptive Test," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 31, no. 9, pp. 1405–1416, Sep. 2012.

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

[237] J. E. Nelson, T. Zanon, J. G. Brown, O. Poku, R. D. Blanton, W. Maly, B. Benware, and C. Schuermyer, "Extracting Defect Density and Size Distributions from Product ICs," *IEEE Design & Test of Computers*, vol. 23, no. 5, pp. 390–400, May 2006.

[238] J. E. Nelson, T. Zanon, R. Desineni, J. G. Brown, N. Patil, W. Maly, and R. D. Blanton, "Extraction of Defect Density and Size Distributions from Wafer Sort Test Results," in *Design, Automation and Test in Europe Conference (DATE)*, vol. 1, March 2006, pp. 1–6.

[239] C. Xue and R. D. Blanton, "Predicting IC Defect Level Using Diagnosis," in *IEEE Asian Test Symposium (ATS)*, Nov 2014, pp. 113–118.

[240] W. C. Tam and S. Blanton, "To DFM or not to DFM?" in *ACM/IEEE Design Automation Conference (DAC)*, June 2011, pp. 65–70.

[241] W. C. Tam and R. D. Blanton, "Design-for-Manufacturability Assessment for Integrated Circuits Using RADAR," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 33, no. 10, pp. 1559–1572, Oct 2014.

[242] R. D. Blanton, F. Wang, C. Xue, P. K. Nag, Y. Xue, and X. Li, "DREAMS: DFM Rule EvAluation using Manufactured Silicon," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2013, pp. 99–106.

[243] R. D. Blanton, F. Wang, C. Xue, P. K. Nag, Y. Xue, and X. Li, "DFM Evaluation Using IC Diagnosis Data," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 36, no. 3, pp. 463–474, March 2017.

[244] L. C. Wagner, *Failure Analysis of Integrated Circuits: Tools and Techniques*. Springer Science & Business Media, 1999, vol. 494.

[245] J. Thomas, J. Baer, P. Westby, K. Mattson, F. Haring, G. Strommen, J. Jacobson, S. S. Ahmad, and A. Reinholz, "A Unique Application of Decapsulation Combining Laser and Plasma," in *Electronic Components and Technology Conference*, May 2009, pp. 2011–2015.

[246] T. W. Lee, "Chemical Decapsulation Revisited," in *International Symposium for Testing and Failure Analysis (ISTFA)*, 1987, pp. 113–122.

[247] D. D. Wilson and J. R. Beall, "Decapsulation of Epoxy Devices using Oxygen Plasma," in *IEEE International Reliability Physics Symposium (IRPS)*, April 1977, pp. 82–84.

[248] D. Thomasi, "Plasma Decapsulation Techniques," *Microelectronic Failure Analysis. Desk Reference, 3rd Edition. ASM International*, pp. 75–78, 1993.

[249] M. Krüger, J. Krinke, K. Ritter, B. Zierle, and M. Weber, "Laser-assisted Decapsulation of Plastic-encapsulated Devices," *Microelectronics Reliability*, vol. 43, no. 9-11, pp. 1827–1831, 2003.

[250] M. H. Hong, Z. Mai, G. Chen, T. Thiam, W. D. Song, Y. Lu, C. E. Soh, and T. C. Chong, "Pulsed Laser Ablation of IC Packages for Device Failure Snalyses," in *Photon Processing in Microelectronics and Photonics*, vol. 4637. International Society for Optics and Photonics, 2002, pp. 445–452.

[251] R. K. Lowry, "Laser Decapsulation Apparatus and Method," U.S. Patent 7,166,186, Jan 23, 2007.

[252] D. Corum, "Mechanical Decap Method for Plastic Devices," in *International Symposium for Testing and Failure Analysis (ISTFA)*, vol. 95, 1984.

[253] L. C. Wagner, "Failure Analysis of Metallization Corrosion," in *ASM Conference on Electronic Packaging*, vol. 275, 1987.

[254] P. Kolodner and J. A. Tyson, "Microscopic Fluorescent Imaging of Surface Temperature Profiles with 0.01 C Resolution," *Applied Physics Letters*, vol. 40, no. 9, pp. 782–784, 1982.

[255] P. Kolodner and J. A. Tyson, "Remote Thermal Imaging with 0.7-μm Spatial Resolution Using Temperature-dependent Fluorescent Thin Films," *Applied Physics Letters*, vol. 42, no. 1, pp. 117–119, 1983.

[256] D. L. Barton, "Fluorescent Microthermographic Imaging," Sandia National Labs., Albuquerque, NM (United States), Tech. Rep., 1993.

[257] D. L. Barton and P. Tangyunyong, "Scanning Fluorescent Microthermal Imaging Apparatus and Method," U.S. Patent 5,705,821, Jan 6, 1998.

[258] E. I. Cole, J. M. Soden, J. L. Rife, D. L. Barton, and C. L. Henderson, "Novel Failure Analysis Techniques using Photon Probing with a Scanning Optical Microscope," in *IEEE International Reliability Physics Symposium (IRPS)*, April 1994, pp. 388–398.

[259] J. A. Rowlette and T. M. Eiles, "Critical Timing Analysis in Microprocessors Using Near-IR Laser Assisted Device Alteration (LADA)," in *IEEE International Test Conference (ITC)*, vol. 1, Sep. 2003, pp. 264–273.

[260] C. F. Hawkins, J. M. Soden, E. I. Cole Jr, and E. S. Snyder, "The Use of Light Emission in Failure Analysis of CMOS ICs," Sandia National Labs., Albuquerque, NM (USA), Tech. Rep., 1990.

[261] K. Wills, T. Lewis, G. Billus, and H. Hoang, "Optical Beam Induced Current Applications for Failure Analysis of VLSI Devices," in *International Symposium for Testing and Failure Analysis (ISTFA)*, vol. 21, 1990.

[262] K. Nikawa and S. Inoue, "Various Contrasts Identifiable From the Backside of a Chip by 1.3μm Laser Beam Scanning and Current Change Imaging," in *International Symposium for Testing and Failure Analysis (ISTFA)*. AMERICAN TECHNICAL PUBLISHERS LTD, 1996, pp. 387–392.

[263] E. I. J. Cole, P. Tangyunyong, and D. L. Barton, "Backside Localization of Open and Shorted IC Interconnections," in *IEEE International Reliability Physics Symposium (IRPS)*, March 1998, pp. 129–136.

[264] A. Gonzales, "On the Electron Beam Induced Current Analysis of Semiconductor Devices," *Scanning Electron Microscopy*, pp. 941–948, 1974.

[265] H. Leamy, "Charge Collection Scanning Electron Microscopy," *Journal of Applied Physics*, vol. 53, no. 6, pp. R51–R80, 1982.

[266] E. I. Cole and R. E. Anderson, "Rapid Localization of IC Open Conductors Using Charge-induced Voltage Alteration (CIVA)," in *Annual Proceedings Reliability Physics*, March 1992, pp. 288–298.

[267] E. Cole, C. Bagnell, B. Davies, A. Neacsu, W. Oxford, and R. Propst, "Advanced Scanning Electron-Microscopy Methods And Applications To Integrated-Circuit Failure Analysis," *Scanning Microscopy*, vol. 2, no. 1, pp. 133–150, 1988.

[268] S. L. Flegler and S. L. Flegler, *Scanning & Transmission Electron Microscopy.* Oxford University Press, 1997.

[269] T. V. Vorburger, J. A. Dagata, G. Wilkening, K. Lizuka, E. Thwaite, and P. Lonardo, "Industrial uses of STM and AFM," *CIRP Annals*, vol. 46, no. 2, pp. 597–620, 1997.

[270] L. Reimer, *Transmission Electron Microscopy: Physics of Image Formation and Microanalysis.* Springer, 2013, vol. 36.

[271] P. Buseck, J. Cowley, and L. Eyring, *High-Resolution Transmission Electron Microscopy and Associated Techniques.* Oxford University Press, 1989.

[272] B. Carter and D. Williams, *Transmission Electron Microscopy.* Springer, 1996, vol. 36.

[273] T. Hattori, "Detection and Identification of Particles on Silicon Surfaces," *Particles on Surfaces, Detection, Adhesion, and Removal, Edited by KL Mittal, Marcel Dekker, Inc., New York*, p. 201, 1995.

[274] S. Mittal and R. D. Blanton, "LearnX: A Hybrid Deterministic-Statistical Defect Diagnosis Methodology," in *IEEE European Test Symposium (ETS)*, May 2019.

[275] S. Mittal and R. D. Blanton, "MD-LearnX: A Deterministic-Statistical Multiple Defect Diagnosis Methodology," in *IEEE VLSI Test Symposium (VTS)*, Apr 2020.

[276] M. Beckler and R. D. Blanton, "Fault Simulation Acceleration for TRAX Dictionary Construction Using GPUs," in *IEEE International Test Conference (ITC)*, Oct 2017, pp. 1–9.

[277] M. Beckler and R. D. Blanton, "GPU-accelerated Fault Dictionary Generation for the TRAX Fault Model," in *IEEE International Test Conference in Asia (ITC-Asia)*, Sep 2017, pp. 34–39.

[278] Z. Stanojevic, H. Balachandran, D. M. H. Walker, F. Lakbani, S. Jandhyala, J. Saxena, and K. M. Butler, "Computer-Aided Fault to Defect Mapping (CAFDM) for Defect Diagnosis," in *IEEE International Test Conference (ITC)*, Oct 2000, pp. 729–738.

[279] J. A. Porche and R. D. Blanton, "Physically-aware Diagnostic Resolution," in *IEEE Asian Test Symposium (ATS)*, Nov 2014, pp. 206–211.

[280] G. Ontko, "Fault Isolation of Large Nets Using Bridging Fault Analysis," in *International Symposium for Testing and Failure Analysis (ISTFA)*, 2004, pp. 99–102.

[281] X. Wen, T. Miyoshi, S. Kajihara, L.-T. Wang, K. K. Saluja, and K. Kinoshita, "On Per-test Fault Diagnosis Using the X-fault Model," in *IEEE International Conference on Computer Aided Design (ICCAD)*, Nov 2004.

[282] Y. Yamato, Y. Nakamura, K. Miyase, X. Wen, and S. Kajihara, "A Novel Per-test Fault Diagnosis Method based on the Extended X-fault model for Deep-submicron LSI Circuits," *IEICE Transactions on Information and Systems*, vol. 91, no. 3, pp. 667–674, 2008.

[283] I. Polian, Y. Nakamura, P. Engelke, S. Spinner, K. Miyase, S. Kajihara, B. Becker, and X. Wen, "Diagnosis of Realistic Defects Based on the X-fault Model," in *IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems*, April 2008.

[284] X. Wen, H. Tamamoto, K. K. Saluja, and K. Kinoshita, "Fault Diagnosis for Physical Defects of Unknown Behaviors," in *IEEE Asian Test Symposium (ATS)*, Nov 2003.

[285] A. P. Dempster, "Upper and Lower Probabilities Induced by a Multivalued Mapping," in *Classic Works of the Dempster-Shafer Theory of Belief Functions*. Springer, 2008, pp. 57–72.

[286] G. Shafer, *A Mathematical Theory of Evidence*. Princeton university press, 1976, vol. 42.

[287] *Modus: Guide 9: Diagnostics*, Product Version 18.11 ed., Cadence Design Systems, Inc, 2018.

[288] *TetraMAX ATPG and TetraMAX II ADV ATPG User Guide*, Version O-2018.06-SP2 ed., Synopsys, Inc, 2018.

[289] *Tessent Diagnosis User's Manual*, Software Version 2019.1 ed., Mentor Graphics Corporation, 2019.

[290] I. Pomeranz, "OBO: An Output-by-output Scoring Algorithm for Fault Diagnosis," in *IEEE Computer Society Annual Symposium on VLSI*, 2014.

[291] K. De and A. Gunda, "Failure Analysis for Full-scan Circuits," in *IEEE International Test Conference (ITC)*, Oct 1995, pp. 636–645.

[292] H. Sabaghian-Bidgoli, P. Behnam, B. Alizadeh, and Z. Navabi, "Reducing Search Space for Fault Diagnosis: A Probability-Based Scoring Approach," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, July 2017, pp. 545–550.

[293] S. Tanwir, S. Prabhu, M. Hsiao, and L. Lingappan, "Information-theoretic and Statistical Methods of Failure Log Selection for Improved Diagnosis," in *IEEE International Test Conference (ITC)*, Oct 2015, pp. 1–10.

[294] Q. Huang, C. Fang, S. Mittal, and R. D. Blanton, "Improving Diagnosis Efficiency via Machine Learning," in *IEEE International Test Conference (ITC)*, Oct 2018.

[295] C. Fang, Q. Huang, S. Mittal, and R. D. Blanton, "Diagnosis Outcome Preview through Learning," in *IEEE VLSI Test Symposium (VTS)*, Apr 2019.

[296] H. Wang, O. Poku, X. Yu, S. Liu, I. Komara, and R. D. Blanton, "Test-data Volume Optimization for Diagnosis," in *IEEE Design Automation Conference (DAC)*, June 2012, pp. 567–572.

[297] H. Wang and K. He, "Improving Test and Diagnosis Efficiency through Ensemble Reduction and Learning," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 24, no. 5, p. 49, 2019.

[298] L. M. Huisman, M. Kassab, and L. Pastel, "Data Mining Integrated Circuit Fails with Fail Commonalities," in *IEEE International Test Conference (ITC)*, Oct 2004, pp. 661–668.

[299] S. Wang and W. Wei, "Machine Learning-based Volume Diagnosis," in *Design, Automation Test in Europe Conference (DATE)*, April 2009, pp. 902–905.

[300] S. Wang, "Machine Learning Based Volume Diagnosis of Semiconductor Chips," U.S. Patent 12/269,380, Jan 7, 2010.

[301] J. E. Nelson, W. C. Tam, and R. D. Blanton, "Automatic Classification of Bridge Defects," in *IEEE International Test Conference (ITC)*, Nov 2010, pp. 1–10.

[302] L. R. Gómez and H. Wunderlich, "A Neural-Network-Based Fault Classifier," in *IEEE Asian Test Symposium (ATS)*, Nov 2016, pp. 144–149.

[303] M. Chern, S.-W. Lee, S.-Y. Huang, Y. Huang, G. Veda, K.-H. H. Tsai, and W.-T. Cheng, "Improving Scan Chain Diagnostic Accuracy Using Multi-stage Artificial Neural Networks," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*. ACM, 2019, pp. 341–346.

[304] Y. Huang, B. Benware, R. Klingenberg, H. Tang, J. Dsouza, and W. Cheng, "Scan Chain Diagnosis Based on Unsupervised Machine Learning," in *IEEE Asian Test Symposium (ATS)*, Nov 2017, pp. 225–230.

[305] B. Benware, C. Schuermyer, M. Sharma, and T. Herrmann, "Determining a Failure Root Cause Distribution From a Population of Layout-Aware Scan Diagnosis Results," *IEEE Design & Test of Computers*, vol. 29, no. 1, pp. 8–18, Feb 2012.

[306] Y. Xue, O. Poku, X. Li, and R. D. Blanton, "PADRE: Physically-Aware Diagnostic Resolution Enhancement," in *IEEE International Test Conference (ITC)*, Sep. 2013, pp. 1–10.

[307] Y. Xue, X. Li, and R. D. Blanton, "Improving Diagnostic Resolution of Failing ICs through Learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. PP, no. 99, 2016.

[308] C. Lim, Y. Xue, X. Li, R. D. Blanton, and M. E. Amyeen, "Diagnostic Resolution Improvement through Learning-guided Physical Failure Analysis," in *IEEE International Test Conference (ITC)*, Nov 2016, pp. 1–10.

[309] L. Breiman, "Random Forests," *Machine Learning*, pp. 5–32, Oct 2001.

[310] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2016.

[311] H. Hsing, "Advanced Encryption Standard (AES)," *https://opencores.org/projects/tiny_aes*, 2012.

[312] I. Parulkar, A. Wood, J. C. Hoe, B. Falsafi, S. V. Adve, J. Torrellas, and S. Mitra, "OpenSPARC: An Open Platform for Hardware Reliability Experimentation," in *Fourth Workshop on Silicon Errors in Logic-System Effects (SELSE)*. Citeseer, 2008.

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

[314] Silvaco Inc., "The Silvaco 45nm Open Cell Library," *https://www.silvaco.com/products/nangate/FreePDK45_Open_Cell_Library*, 2011.

[315] R. D. Blanton, J. T. Chen, R. Desineni, K. N. Dwarakanath, W. Maly, and T. J. Vogels, "Fault Tuples in Diagnosis of Deep-submicron Circuits," in *IEEE International Test Conference (ITC)*, Oct 2002, pp. 233–241.

[316] R. D. Blanton, K. N. Dwarakanath, and R. Desineni, "Defect Modeling Using Fault Tuples," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 25, no. 11, pp. 2450–2464, Nov 2006.

[317] K. N. Dwarakanath and R. D. Blanton, "Universal Fault Simulation Using Fault Tuples," in *IEEE Design Automation Conference (DAC)*. ACM, 2000, pp. 786–789.

[318] P. Chen, C. Che, J. C. . Li, S. Kuo, P. Hsueh, C. Kuo, and J. Lee, "Physical-aware Systematic Multiple Defect Diagnosis," *IET Computers Digital Techniques*, vol. 8, no. 5, pp. 199–209, Sep. 2014.

[319] S. Chao, P. Chen, J. Chen, P. Chen, A. Lin, J. C. . Li, P. Hsueh, C. Kuo, Y. Chen, and J. Li, "Divide and Conquer Diagnosis for Multiple Defects," in *IEEE International Test Conference (ITC)*, Oct 2014, pp. 1–8.

[320] Z. Wang, K. Tsai, M. Marek-Sadowskal, and J. Rajski, "An Efficient and Effective Methodology on the Multiple Fault Diagnosis," in *IEEE International Test Conference (ITC)*, vol. 1, Sep. 2003, pp. 329–338.

[321] P. Chen, C. Lee, J. Chen, P. Chen, and J. C. Li, "Physical-aware Diagnosis of Multiple Interconnect Defects," in *International Test Conference in Asia (ITC-Asia)*, Sep. 2017.

[322] X. Tang, W. Cheng, R. Guo, and S. M. Reddy, "Diagnosis of Multiple Physical Defects using Logic Fault Models," in *IEEE Asian Test Symposium (ATS)*, Dec 2010.

[323] P.-Y. Hsueh, C.-Y. Kuo, C.-M. Li, and C.-C. Che, "Multiple Defect Diagnosis Method and Machine Readable Media," U.S. Patent 9,983,264, May 29, 2018.

[324] A. Ladhar and M. Masmoudi, "Analysis and Diagnosis of Multiple Simultaneous Defects," in *IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Dec 2009, pp. 671–674.

[325] A. Ladhar, I. Izaute, and M. Masmoudi, "A New Approach to Diagnose Multiple Simultaneous Defects," in *International Conference on Signals, Circuits and Systems*, Nov 2009, pp. 1–6.

[326] Z. Wang, M. Marek-Sadowska, K. Tsai, and J. Rajski, "Analysis and Methodology for Multiple-fault Diagnosis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 25, no. 3, pp. 558–575, March 2006.

[327] J. B. Liu and A. Veneris, "Incremental Fault Diagnosis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 24, no. 2, pp. 240–251, Feb 2005.

[328] A. Veneris, J. B. Liu, M. Amiri, and M. S. Abadir, "Incremental Diagnosis and Correction of Multiple Faults and Errors," in *Design, Automation and Test in Europe Conference (DATE)*, March 2002, pp. 716–721.

[329] J. B. Liu, A. Veneris, and H. Takahashi, "Incremental Diagnosis of Multiple Open-interconnects," in *IEEE International Test Conference (ITC)*, Oct 2002, pp. 1085–1092.

[330] Shi-Yu Huang, "On Improving the Accuracy of Multiple Defect Diagnosis," in *IEEE VLSI Test Symposium (VTS)*, April 2001, pp. 34–39.

[331] Shi-Yu Huang, Kwang-Ting Cheng, Kuang-Chien Chen, and D. I. Cheng, "ErrorTracer: A Fault Simulation-based Approach to Design Error Diagnosis," in *IEEE International Test Conference (ITC)*, Nov 1997, pp. 974–981.

[332] Shi-Yu Huang and Kwang-Ting Cheng, "ErrorTracer: Design Error Diagnosis Based on Fault Simulation Techniques," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 18, no. 9, pp. 1341–1352, Sep. 1999.

[333] S. Kundu, S. Chattopadhyay, I. Sengupta, and R. Kapur, "Multiple Fault Diagnosis Based on Multiple Fault Simulation Using Particle Swarm Optimization," in *IEEE Internatioal Conference on VLSI Design (VLSID)*, Jan 2011, pp. 364–369.

[334] S. Kundu, A. Jha, S. Chattopadhyay, I. Sengupta, and R. Kapur, "Framework for Multiple-Fault Diagnosis Based on Multiple Fault Simulation Using Particle Swarm Optimization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 3, pp. 696–700, March 2014.

[335] J. Anita and P. Vanathi, "Multiple Fault Diagnosis and Test Power Reduction Using Genetic Algorithms," in *International Conference on Eco-friendly Computing and Communication Systems*. Springer, 2012, pp. 84–92.

[336] J. P. Anita and P. T. Vanathi, "Genetic algorithm based test pattern generation for multiple stuck-at faults and test power reduction in vlsi circuits," in *International Conference on Electronics and Communication Systems (ICECS)*, Feb 2014, pp. 1–6.

[337] J. Ye, Y. Hu, X. Li, W. Cheng, Y. Huang, and H. Tang, "Diagnose Failures Caused by Multiple Locations at a Time," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 4, pp. 824–837, April 2014.

[338] T. G. Nath, E. Midhila, A. Swaminathan, B. Lekshmi, and J. Anita, "Diagnosis of Multiple Stuck-at Faults Using Fault Element Graph with Reduced Power," in *International Symposium on Security in Computing and Communication*. Springer, 2016, pp. 414–426.

[339] X. Tang, W. Cheng, R. Guo, H. Tang, and S. M. Reddy, "Diagnosis of Multiple Faults Based on Fault-Tuple Equivalence Tree," in *IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, Oct 2011, pp. 217–225.

[340] J. Ye, Y. Hu, and X. Li, "Diagnosis of Multiple Arbitrary Faults with Mask and Reinforcement Effect," in *Design, Automation Test in Europe Conference (DATE)*, March 2010, pp. 885–890.

[341] X. Yu and R. D. Blanton, "Multiple Defect Diagnosis Using no Assumptions on Failing Pattern Characteristics," in *ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2008, pp. 361–366.

[342] X. Yu and R. D. Blanton, "An Effective and Flexible Multiple Defect Diagnosis Methodology Using Error Propagation Analysis," in *IEEE International Test Conference (ITC)*, Oct 2008, pp. 1–9.

[343] X. Yu and R. D. Blanton, "Diagnosis of Integrated Circuits With Multiple Defects of Arbitrary Characteristics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 29, no. 6, pp. 977–987, June 2010.

[344] M. Tasi, M. C. T. Chao, J. Jou, and M. Wu, "Multiple-Fault Diagnosis Using Faulty-Region Identification," in *IEEE VLSI Test Symposium (VTS)*, May 2009, pp. 123–128.

[345] V. Boppana, R. Mukherjee, J. Jain, M. Fujita, and P. Bollineni, "Multiple Error Diagnosis Based on Xlists," in *IEEE Design Automation Conference (DAC)*, June 1999, pp. 660–665.

[346] V. Boppana, R. Mukherjee, J. Jain, and M. Fujita, "Multiple Error and Fault Diagnosis Based on Xlists," U.S. Patent 6,532,440, Mar 11, 2003.

[347] N. Sridhar and M. S. Hsiao, "On Efficient Error Diagnosis of Digital Circuits," in *IEEE International Test Conference (ITC)*, Nov 2001, pp. 678–687.

[348] A. L. D'Souza and M. S. Hsiao, "Error Diagnosis of Sequential Circuits Using Region-based Model," in *International Conference on VLSI Design (VLSID)*, Jan 2001, pp. 103–108.

[349] A. L. D'Souza and M. S. Hsiao, "Error Diagnosis of Sequential Circuits Using Region-based Model," *Journal of Electronic Testing*, vol. 21, no. 2, pp. 115–126, 2005.

[350] I. Pomeranz, "Test Scores for Improving the Accuracy of Logic Diagnosis for Multiple Defects," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 7, pp. 1720–1724, July 2019.

[351] Y. Lim, J. Park, and S. Kang, "A Method for the Fast Diagnosis of Multiple Defects Using an Efficient Candidate Selection Algorithm," *IEICE Electronics Express*, vol. 9, no. 9, pp. 834–839, 2012.

[352] Lixing Zhao and V. D. Agrawal, "Net Diagnosis Using Stuck-at and Transition Fault Models," in *IEEE VLSI Test Symposium (VTS)*, April 2012, pp. 221–226.

[353] H. Stratigopoulos, "Machine Learning Applications in IC Testing," in *IEEE European Test Symposium (ETS)*, May 2018.

[354] M. Renovell, P. Huc, and Y. Bertrand, "The Concept of Resistance Interval: A New Parametric Model for Realistic Resistive Bridging Fault," in *IEEE VLSI Test Symposium (VTS)*, April 1995, pp. 184–189.

[355] K. Chow, "Electromigration Protection Requires Accurate Interconnect Modeling," *https://blogs.mentor.com/calibre/blog/2016/08/22/electromigration-protection-requires-accurate-interconnect-modeling/*, 2016.

[356] K. Chow, "The Challenges and Impact of Parasitic Extraction at 65 nm," in *International Symposium on Quality Electronic Design (ISQED)*. IEEE Computer Society, 2006, pp. 697–702.

[357] C. Mack, *Fundamental Principles of Optical Lithography: The Science of Microfabrication*. John Wiley & Sons, 2008.

[358] D. Arumi, R. Rodriguez-Montanes, J. Figueras, S. Eichenberger, C. Hora, and B. Kruseman, "Gate Leakage Impact on Full Open Defects in Interconnect Lines," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 12, pp. 2209–2220, Dec 2011.

[359] A. Raghunathan, G. Fenger, M. Lam, C. Clifford, K. Adam, and J. Sturtevant, "Edge Placement Errors in EUV from Aberration Variation," in *Photomask Technology 2017*, vol. 10451. International Society for Optics and Photonics, 2017, pp. 221–229.

[360] *International Technology Roadmap for Semiconductors*, 2009th ed., Semiconductor Industry Association, Austin, TX, 2009.

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

[362] W. C. Tam and R. D. Blanton, "SLIDER: A Fast and Accurate Defect Simulation Framework," in *IEEE VLSI Test Symposium (VTS)*, May 2011, pp. 172–177.

[363] W. C. Tam and R. D. Blanton, "SLIDER: Simulation of Layout-Injected Defects for Electrical Responses," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 31, no. 6, pp. 918–929, June 2012.

[364] A. Strojwas, "Personal communication."

[365] Z. Stanojevic and D. M. H. Walker, "FedEx - A Fast Bridging Fault Extractor," in *IEEE International Test Conference (ITC)*, Nov 2001, pp. 696–703.

[366] A. Jee and F. J. Ferguson, "Carafe: An Inductive Fault Analysis Tool for CMOS VLSI circuits," in *IEEE VLSI Test Symposium (VTS)*, April 1993, pp. 92–98.

[367] F. J. Ferguson and J. P. Shen, "A CMOS Fault Extractor for Inductive Fault Analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 7, no. 11, pp. 1181–1194, Nov 1988.

[368] L. Ko, S. Huang, J. Chiou, and H. Cheng, "Modeling and Testing of Intra-Cell Bridging Defects Using Butterfly Structure," in *International Symposium on VLSI Design, Automation and Test*, April 2006, pp. 1–4.

[369] A. Ladhar and M. Masmoudi, "Extraction and Simulation of Intra-gate Defects Affecting CMOS Libraries," *Journal of Computers*, vol. 5, pp. 1468–1477, 2010.

[370] Y. Liu and Q. Xu, "On Modeling Faults in FinFET Logic Circuits," in *IEEE International Test Conference (ITC)*, Nov 2012, pp. 1–9.

[371] K. Chiang, Y. Ho, Y. Chen, C. Pan, and J. C. Li, "Fault Simulation and Test Pattern Generation for Cross-gate Defects in FinFET Circuits," in *IEEE Asian Test Symposium (ATS)*, Nov 2015, pp. 181–186.

[372] S. Zachariah, Y.-S. Chang, S. Kundu, and C. Tirumurti, "On Modeling Cross-Talk Faults," in *Design, Automation and Test in Europe Conference (DATE)*, Mar 2003, pp. 490–495.

[373] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "ASAP7: A 7-nm FinFET Predictive Process Design Kit," *Microelectronics Journal*, vol. 53, pp. 105–115, Jul 2016.

[374] Osei Poku and R. D. Blanton, "Delay Defect Diagnosis using Segment Network Faults," in *IEEE International Test Conference (ITC)*, Oct 2007, pp. 1–10.