A Deterministic-Statistical Multiple-Defect Diagnosis Methodology

Soumya Mittal and R. D. (Shawn) Blanton
Advanced Chip Test Laboratory
Department of Electrical and Computer Engineering
Carnegie Mellon University, Pittsburgh, PA 15213

***Abstract*—Software diagnosis is the 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. However, multiple-defect diagnosis is challenging due to error masking and unmasking effects, and exponential complexity of the solution search process. This paper describes a three-phase, physically-aware diagnosis methodology called MD-LearnX to effectively diagnose multiple defects, and in turn, aid in accelerating the design and process development. The first phase identifies a defect that resembles traditional fault models. The second and the third phases utilize the X-fault model and machine learning to identify correct candidates. Results from a thorough fault injection and simulation experiment demonstrate that MD-LearnX returns an ideal diagnosis 2X more often than commercial diagnosis. Its effectiveness is further evidenced through a silicon experiment, where, on average, MD-LearnX returns 5.3 fewer candidates per diagnosis as compared to state-of-the-art commercial diagnosis without losing accuracy.**

I. Introduction

As semiconductor fabrication advances to smaller process nodes, it is becoming increasingly difficult to ramp yield quickly. Advanced nodes introduce new design and manufacturing challenges, resulting in an increase in the number and complexity of defect mechanisms, which consequently hinder yield learning. The rate of yield learning dictates the success of a new fabrication process, design, etc., and thus must be accelerated to meet diminishing time-to-market and time-to-volume requirements.

Software-based defect diagnosis plays an important role in failure analysis. 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. Diagnosis is therefore an indispensable tool for facilitating yield learning.

Diagnosis methods in the literature can essentially be differentiated based on the type of the fault model used (the (temporary) stuck-at fault model [1]–[10] vs. the X-fault model [11]–[15]), the scoring technique employed (deterministic [3]–[16] vs. statistical [17]), how precisely a defect is localized (i.e., whether a defect candidate is reported at a logic, back-end layout [18] or front-end layout level [19]), and whether multiple defects affecting a single chip can be analyzed/identified. [4]–[12].

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

To demonstrate 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 1.

However, 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). Second, the solution search space is exponential in defect multiplicity (unknown beforehand), which makes finding an optimum solution extremely difficult.

Numerous methods have been suggested over the years to diagnose multiple defects. The first category of methods [6]–[10] rely on identifying a defect via failing patterns (a) that can be explained by a single location, and (b) where errors propagating from multiple locations do not interfere with each other. However, such patterns can be limited and/or find an incorrect location due to a considerable number of interactions

TABLE 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.

Chip type	Process node (nm)	Failing chips diagnosed with multiple defects (%)	No. of failing chips
Test chip	110	7.7	5416
High-volume chip	55	11.7	1201
Test chip	14	15.6	1375
Test chip	28	29.8	1952
Test chip	14	32.2	11727
High-volume chip	130	32.6	328
High-volume chip	90	35.3	11196
High-volume chip [8]	130	41.0	453
High-volume chip	–	49.8	353
Test chip	28	53.9	167
[6]	55	60.0	209

among errors manifesting from multiple defect locations. The second category of methods [1], [2] focus on finding each defect incrementally but involve a significant number of multiple-fault simulations, and hence do not scale well with design size and defect multiplicity. The third category of methods (e.g., [20]), that guide their effort in exploring the exponential search space via optimization techniques, face a similar drawback and are thus impractical. The fourth category of methods [4], [11] avoid explicit fault simulation to identify multiple defects. 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.

Another way to avoid the inherent problem of error masking and unmasking in multiple-defect diagnosis is to employ the X-fault model [12]–[15]. The X-fault model assumes an unknown (X) value at a potential defect location and allows error to propagate conservatively.

Furthermore, prior work discussed up to this point uses various candidate-ranking heuristics to identify the best set of defect locations that can represent actual defects. However, such heuristics are explicitly created based on intuition and domain knowledge, and are thus 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 [17]. Thus, an alternative to rank candidates is machine learning.

Machine learning (ML) has been successfully applied in chip testing [21]. Specifically, in the area of diagnosis, ML has been used to optimize test data collection to make diagnosis more efficient [22], [23], improve the accuracy and resolution of diagnosis itself [17], and pinpoint yield-limiting layout geometries by analyzing a volume of diagnosis data [24], [25].

Recent work introduced a diagnosis methodology called LearnX [17] that employs the X-fault model to avoid eliminating an actual defect location, and machine learning to identify the best candidate that could represent a defect. LearnX achieves superior performance for single-defect diagnosis when compared with a state-of-the-art commercial tool; it returns a single correct candidate for 86.6% more fail logs.

This work describes a single-chip diagnosis methodology that we term MD-LearnX. It builds on LearnX to effectively tackle the task of characterizing more than one defect and addresses the drawbacks of prior work related to multiple-defect diagnosis discussed up to this point. 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, it utilizes machine learning to create a scoring model that learns the latent correlations between a correct candidate and the tester response.

3. It uses design layout information to identify the physical defect type and behavior of each candidate. Back-end and front-end layout analysis techniques [18], [19] further strengthen MD-LearnX by improving its physical resolution.

It should be noted that an approach that improves the quality of diagnosis by either reordering, selecting or adding test patterns complement MD-LearnX [26], [27].

The rest of the paper is organized as follows. Section 2 provides a detailed overview of MD-LearnX. Section 3 describes several experiments that demonstrate the effectiveness of MD-LearnX. The final section concludes this paper.

II. Diagnosis Methodology

Fig. 1 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. The first few steps common in each phase are discussed in Section II-A. The first phase (Section II-B) 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 II-C) 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 referred to as class-2 defects. Finally, any failing chip left undiagnosed is analyzed in Phase 3 (Section II-D). 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 2 summarizes the class and multiplicity of defects targeted by each phase of MD-LearnX.

A. *Phase* 1-3 *steps*

MD-LearnX begins with path tracing [28], where it traces back from each failing output for each failing pattern to infer potential defective lines. Path tracing guarantees that it will

TABLE 2
Class and multiplicity of defects targeted by each phase of MD-LearnX.

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

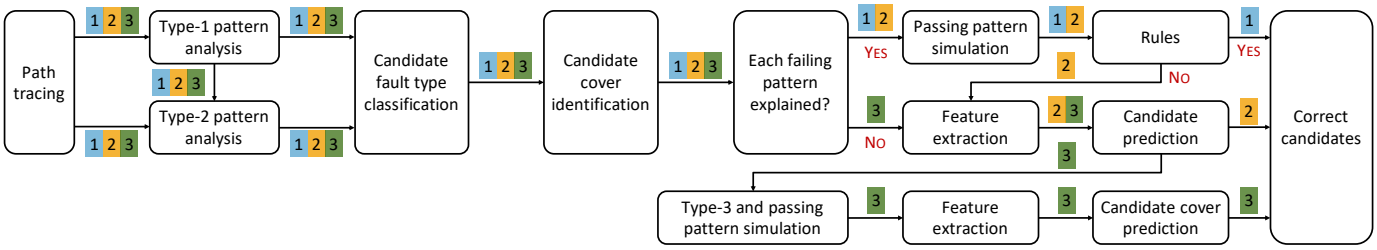

Fig. 1. Overview of the proposed three-phase diagnosis methodology, MD-LearnX. The sequence of steps in each phase are marked with a different color.

find a defect location that assists in error propagation to at least one design output, even for multiple defects.

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 with respect to a (temporary) stuck-at fault, not an X fault, in the literature [7], [8]. However, as discussed in Section I, 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 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 defect types considered in this work are STUCK, CELL, BRIDGE and OPEN.

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 II-B and II-C, respectively.

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

B. *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 (Table 3) are constructed for each candidate defect 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). 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 virtue of Occam's Razor).

C. *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.

Specifically, a commonly used supervised machine learning algorithm called a random forest [29] 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 [17]. The features are derived from the stuck-at and the X fault simulation of a defect candidate, and likely, completely, represents its behavior.

For each defect 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 multiple defects in the

TABLE 3
RULES FOR EACH CANDIDATE DEFECT TYPE IN PHASE 1.

Defect type	Rule
STUCK	Candidate passes for each passing pattern.
OPEN	Candidate passes for each passing pattern.
CELL	Cell passes consistency check [3].
BRIDGE	Bridged nets have opposite polarities for failing patterns. Bridged nets have the same polarity for passing patterns that sensitize one of the nets to a design output.

Common rule: Candidate explains each failing pattern.

circuit. Hyperparameters of each model are optimized using a separate validation dataset.

Because only a single candidate can represent an actual defect, the training and the validation datasets are highly imbalanced. An optimum decision threshold for each trained model is thus selected using the Precision-Recall curve [17], [30]. Each defect candidate (according to its defect type) is then adjudged correct/incorrect by the corresponding trained ML model.

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.

D. Phase 3

If there is 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, ML classification is applied at two different levels; first, at a defect level, where defect-type specific ML models built using the approach outlined in Phase 2 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 and passing patterns. Thus, 32 features that correspond to the failing patterns are used to train a defect-level ML model. Second, 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 ML model. Each cover is then simulated for the remaining patterns (type-3 and passing patterns). Here, machine learning is utilized at a candidate cover level. A single random forest is trained using the steps similar to that described in Phase 2, with the difference being that each training/validation instance is a candidate cover in Phase 3. A cover is then said to represent actual defects when it is predicted correct by the ML model.

It should be noted that unlike [1], [2], multiple-fault simulation is performed here to extract features, and not explore the exponential search space. Unlike the work of [5]–[10], type-3 patterns are used here to further improve the quality of diagnosis. Unlike [5], MD-LearnX can effectively handle multiple byzantine defects [31] without exhaustive enumeration of all fault combinations at a possible open defect location.

III. Experiment

Two experiments are described to validate MD-LearnX: one is a fault injection and simulation experiment using three different designs (Section III-A), and another that uses silicon failure data (Section III-B).

A. Simulation

A simulation-based experiment is performed using three designs – one is an Advanced Encryption Standard (AES) core that provides AES-128 encryption, the second design is the L2B cache write-back buffer (called L2B) of the OpenSPARC T2 processor [32], and the third design is an ITC'99 benchmark called B18. Realistic physical defect behaviors associated with bridge, open and cell defects (including byzantine bridges [33] and opens [31]) are modeled here [17]. For each design and defect multiplicity, 1,000 virtual fail logs are generated by uniquely and randomly selecting the defect locations and their behaviors. For each design, additional 500 fail logs are used to produce the training dataset and another 500 to produce the validation dataset, 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 diagnosis quality is assessed using three metrics, namely, diagnosability, resolution and *home run*. Diagnosability is defined as the ratio of the number of defect locations that are correctly identified to the number of defect locations injected [11]. Resolution is defined as the ratio of the number of defect locations that are correctly identified to the number of defect locations reported. A diagnosis approach is said to hit a home run when diagnosability and resolution are simultaneously equal to one (i.e., perfect diagnosis).

Fig. 2(a) compares the average diagnosability (y-axis) achieved by MD-LearnX with commercial diagnosis for the AES core for various defect multiplicities (x-axis). Fig. 2(a) clearly shows that MD-LearnX performs better than commercial diagnosis. Fig. 2(a) reveals that the average diagnosability of MD-LearnX is 0.77, which is 18.4% (1.1X) more than Tool 1 (Tool 2). When a circuit is affected by five or more defects, MD-LearnX performs even better, where the enhancement over Tool 1 (Tool 2) is 23.5% (1.4X). Further analysis indicates that MD-LearnX correctly identifies all the injected defect locations (i.e., diagnosability = 1) for 52.9% fail logs, which is 22.2% (2.5X) higher than Tool 1 (Tool 2).

Fig. 2(b) compares the average resolution (y-axis) attained by MD-LearnX with commercial diagnosis for various defect multiplicities (x-axis). Fig. 2(b) shows that the average resolution of MD-LearnX is 0.67, which is 97.0% (1.2X) more than Tool 1 (Tool 2). It performs even better when a circuit is affected by five or more defects; its resolution is 2.3X (2.4X) times Tool 1 (Tool 2). Further analysis reveals that MD-LearnX achieves ideal resolution for 3X as many fail logs as commercial diagnosis.

It is observed from Fig. 2(c) that MD-LearnX delivers a home run for 26.1% fail logs, which is 2.9X (4X) times Tool 1 (Tool 2). For defect multiplicity more than 5, where commercial diagnosis nearly does not return an ideal diagnosis, MD-LearnX hits a home run for 4.3% fail logs.

The improvement in the quality of diagnosis achieved by MD-LearnX over commercial diagnosis for the three designs examined is summarized in Table 4. On average, the diagnos-

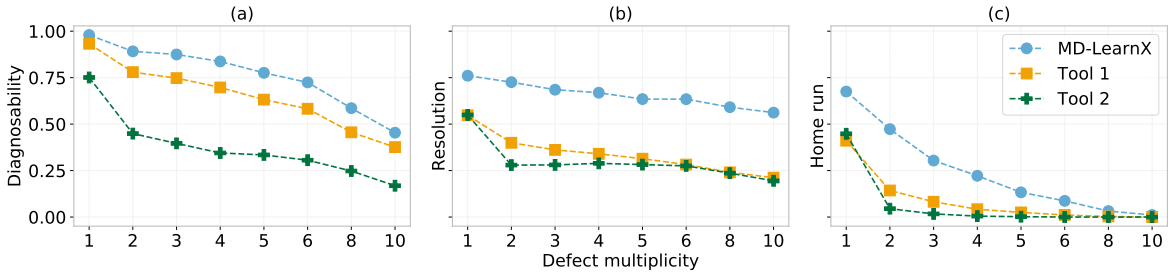

Fig. 2. (a) Diagnosability, (b) resolution and (c) *home run* of MD-LearnX compared with commercial diagnosis for the AES core.

TABLE 4
DIAGNOSIS QUALITY IMPROVEMENT OVER COMMERCIAL DIAGNOSIS FOR DIFFERENT DESIGNS.

Design	Diagnosability (%)		Resolution (%)		*Home run* (%)	
	Tool 1	Tool 2	Tool 1	Tool 2	Tool 1	Tool 2
AES	18.5	108.1	97.1	123.3	188.9	271.4
L2B	26.0	173.9	44.2	341.2	75.0	966.7
B18	32.0	127.6	59.0	129.6	340.0	633.3
Average	25.5	136.5	66.8	198.0	201.3	623.8

ability of MD-LearnX is 25.5% (1.4X) more than Tool 1 (Tool 2), and the resolution is 66.8% (2X) higher than Tool 1 (Tool 2). Additionally, the number of home runs hit by MD-LearnX is three (seven) times Tool 1 (Tool 2), on average.

Furthermore, MD-LearnX is evaluated based on its ability to estimate the defect multiplicity. Fig. 3 shows the box-plot distribution of the number of defects estimated (y-axis) by MD-LearnX and commercial diagnosis for each injected defect multiplicity (x-axis).

Fig. 3 expectedly reveals that as defect multiplicity increases, the likelihood of misprediction increases. For example, MD-LearnX correctly estimates the number of defects for 65.2% of fail logs when the actual defect multiplicity is two, compared to 47.5% (53.5%) by Tool 1 (Tool 2); when the number of injected defects is 10, MD-LearnX correctly predicts for 9.0% of fail logs, compared to 7.9% (1.3%) by Tool 1 (Tool 2). On average, MD-LearnX correctly determines the defect multiplicity for 36.7% fail logs, which is 37.5% (20.8%) more than Tool 1 (Tool 2).

When the runtime (wall clock time or elapsed real time, to be precise) of MD-LearnX is compared with commercial diagnosis, it is observed that, on average, MD-LearnX is

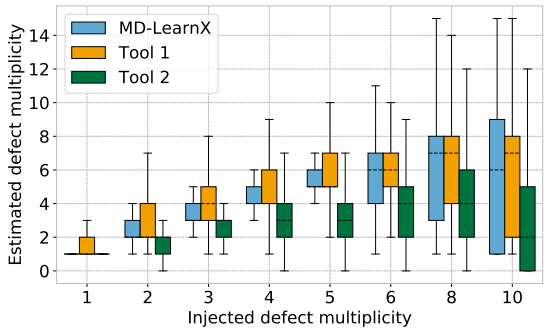

Fig. 3. Distribution of defect multiplicity estimated by MD-LearnX and commercial diagnosis for the AES core.

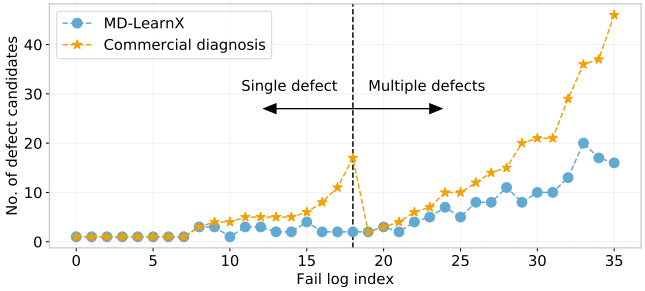

Fig. 4. Number of candidates returned by MD-LearnX for 36 silicon fail logs.

25.2% slower than Tool 1 and 59.7% faster than Tool 2. This comparison, however, 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) rather than the runtime efficiency of MD-LearnX.

B. Silicon

MD-LearnX is also applied to actual failing chips to evaluate its performance. The silicon failure data comes from a 40M gate test chip design manufactured in 14nm technology. For each fail log, it is seen that MD-LearnX (and commercial diagnosis) correctly localizes the defects that are PFAed. More importantly, MD-LearnX returns fewer candidates than commercial diagnosis, on average, without sacrificing accuracy.

Fig. 4 compares the number 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[2]. Fig. 4 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%. 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.

To summarize, MD-LearnX is shown to be effective in a simulation-based and a silicon experiment, with significant enhancement over state-of-the-art commercial diagnosis.

[2]The available silicon design and test data conforms to only one of the commercial diagnosis tools (Tool 1).

IV. Conclusions

In this work, a single-chip diagnosis methodology called MD-LearnX is described to effectively diagnose multiple defects. It is a physically-aware, three-phase diagnosis methodology. Phase 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 defect type) that learns the hidden correlations between the tester response and the correct candidate is created to pinpoint 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 21,000 faulty circuits with varying defect multiplicities and behaviors are created and analyzed. The proposed methodology achieves an average diagnosability and resolution of 0.69 and 0.68, respectively, an improvement of 25.5% and 66.8% over commercial diagnosis. In addition, MD-LearnX delivers a home run, i.e., returns an ideal diagnosis, for 26.7% of the fail logs, which is twice as many as the better-of-the-two commercial diagnosis tools.

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.

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. Future work focuses on extracting design- and layout-specific features in Phase 2 and Phase 3 to further improve the performance of MD-LearnX.

References

[1] Zhiyuan 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*, vol. 25, no. 3, pp. 558–575, March 2006.

[2] J. B. Liu and A. Veneris, "Incremental Fault Diagnosis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 2, pp. 240–251, Feb 2005.

[3] 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*, Oct 2006.

[4] J. Ye *et al.*, "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.

[5] P. Chen *et al.*, "Physical-aware Diagnosis of Multiple Interconnect Defects," in *International Test Conference in Asia (ITC-Asia)*, Sep. 2017.

[6] ——, "Physical-aware Systematic Multiple Defect Diagnosis," *IET Computers Digital Techniques*, vol. 8, no. 5, pp. 199–209, Sep. 2014.

[7] X. Tang, W. Cheng, R. Guo, and S. M. Reddy, "Diagnosis of Multiple Physical Defects using Logic Fault Models," in *IEEE Asian Test Symposium*, Dec 2010.

[8] 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*, Nov 2001.

[9] S. Holst and H. J. Wunderlich, "Adaptive Debug and Diagnosis without Fault Dictionaries," in *IEEE European Test Symposium*, May 2007.

[10] 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*, Oct 2002.

[11] 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*, vol. 29, no. 6, pp. 977–987, June 2010.

[12] V. Boppana *et al.*, "Multiple Error Diagnosis based on Xlists," in *Design Automation Conference*, June 1999, pp. 660–665.

[13] V. Boppana and M. Fujita, "Modeling the Unknown! Towards Model-independent Fault and Error Diagnosis," in *IEEE International Test Conference*, Oct 1998, pp. 1094–1101.

[14] X. Wen *et al.*, "On Per-test Fault Diagnosis Using the X-fault model," in *IEEE International Conference on Computer Aided Design*, Nov 2004.

[15] I. Polian *et al.*, "Diagnosis of Realistic Defects Based on the X-fault Model," in *IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems*, April 2008.

[16] I. Pomeranz, "OBO: An Output-by-output Scoring Algorithm for Fault Diagnosis," in *IEEE Computer Society Annual Symposium on VLSI*, 2014.

[17] S. Mittal and R. D. Blanton, "LearnX: A Hybrid Deterministic-Statistical Defect Diagnosis Methodology," in *IEEE European Test Symposium*, May 2019.

[18] ——, "PADLOC: Physically-aware Defect Localization and Characterization," in *IEEE Asian Test Symposium*, Nov 2017.

[19] ——, "NOIDA: Noise-resistant Intra-cell Diagnosis," in *IEEE VLSI Test Symposium*, April 2018.

[20] S. Kundu *et al.*, "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.

[21] H. Stratigopoulos, "Machine Learning Applications in IC Testing," in *IEEE European Test Symposium (ETS)*, May 2018.

[22] Q. Huang, C. Fang, S. Mittal, and R. D. Blanton, "Improving Diagnosis Efficiency via Machine Learning," in *IEEE International Test Conference*, Oct 2018.

[23] C. Fang, Q. Huang, S. Mittal, and R. D. Blanton, "Diagnosis Outcome Preview through Learning," in *IEEE VLSI Test Symposium*, Apr 2019.

[24] W. Cheng *et al.*, "Automatic Identification of Yield Limiting Layout Patterns Using Root Cause Deconvolution on Volume Scan Diagnosis Data," in *IEEE Asian Test Symposium*, Nov 2017.

[25] R. D. Blanton *et al.*, "Yield Learning through Physically Aware Diagnosis of IC-failure Populations," *IEEE Design Test of Computers*, vol. 29, no. 1, pp. 36–47, Feb 2012.

[26] N. Wang *et al.*, "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*, Oct 2018.

[27] 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.

[28] S. Venkataraman and W. K. Fuchs, "A Deductive Technique for Diagnosis of Bridging Faults," in *IEEE International Conference on Computer Aided Design*, Nov 1997, pp. 562–567.

[29] L. Breiman, "Random Forests," *Machine Learning*, pp. 5–32, Oct 2001.

[30] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2016.

[31] S.-Y. Huang, "Diagnosis of Byzantine Open-segment Faults," in *IEEE Asian Test Symposium*, Nov 2002, pp. 248–253.

[32] I. Parulkar *et al.*, "OpenSPARC: An Open Platform for Hardware Reliability Experimentation," in *Workshop on Silicon Errors in Logic-System Effects*, 2008.

[33] 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*, Oct 1993.