LearnX: A Hybrid Deterministic-Statistical Defect Diagnosis Methodology

Soumya Mittal
Advanced Chip Test Laboratory
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, USA

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

***Abstract*—Software-based diagnosis analyzes the observed response of a failing circuit to pinpoint potential defect locations and deduce their respective behaviors. It plays a crucial role in finding the root cause of failure, and subsequently facilitates yield analysis, learning and optimization. This paper describes a two-phase, physically-aware diagnosis methodology called LearnX to improve the quality of diagnosis, and in turn the quality of design, test and manufacturing. The first phase attempts to diagnose a defect that manifests as a well-established fault behavior (e.g., stuck 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. Results from 30,000 fault injection experiments indicate that LearnX returns an ideal diagnosis result (i.e., a single candidate correctly representing the injected fault) for 73.2% of faulty circuits, which is 86.6% higher than state-of-the-art commercial diagnosis. Silicon experiments further demonstrate the value of LearnX.**

I. Introduction

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. The rate of yield learning is extremely critical to the success of the semiconductor industry and must thus be accelerated to meet the triple objectives of diminishing time-to-volume, time-to-market and time-to-money requirements.

Various strategies are used to identify and characterize the sources of yield loss. Inline inspection, for example, optically examines a wafer to characterize defects. However, with technology continuing to shrink, its effectiveness to locate a defect further decreases [1]. Specialized test structures such as comb drives and ring oscillators are transparent to failure but do not reflect the diversity of the layout patterns found in an actual customer chip. More importantly, actual chips undergo additional fabrication steps that may introduce defect mechanisms that are simply not possible in simple test structures. Hence, in recent years, the use of legacy designs retrofitted for the latest technology node (and logic test chips [2]) have been gaining traction as yield learning vehicles [1], [3], [4].

Specifically, the knowledge of how a customer chip fails manufacturing test is used to improve yield. This process is called failure analysis (FA). FA typically starts with software-based diagnosis, which is the process of identifying the location and behavior of a defect in a failed chip by analyzing the observed circuit response. The quality of diagnosis is generally measured through two metrics; namely, resolution, which is defined as the number of defect candidates reported, and accuracy, defined as whether any reported candidate correspond to an actual defect.

The next step in FA can be volume diagnosis [3], [4], where diagnosis results for a population of failing chips are correlated to find yield-limiting design/manufacturing issues, in hope of determining if a significant percentage of chips are failing due to a common root cause. Chips with yield-limiting defects can then be examined using physical failure analysis (PFA). PFA is the process of visually inspecting a chip for defects and provides indisputable confirmation of the presence of a defect. Thus, the feedback obtained from PFA is used to understand failure mechanisms, and amend the design and/or the manufacturing process to improve yield.

In addition to guiding PFA, volume diagnosis can provide useful information to improve test and diagnosis. For example, it can be used to estimate defect-type distribution, which can then be used to reduce test escapes via adaptive testing [4]. In [5], diagnosis data is used to find the efficiency of new and existing test methods, instead of relying on time-consuming test-set silicon experiments.

High diagnosis accuracy and resolution are thus extremely important for improving design, test and manufacturing of a chip. An inaccurate diagnosis, for example, can direct volume diagnosis to find incorrect correlations within diagnosis results, which in turn can steer PFA to inspect incorrect die locations. The destructive and time-consuming nature of PFA constrains it to being performed on only a small number of chips and considerable amount of resources can be wasted if the quality of diagnosis is poor.

A typical diagnosis algorithm begins with analyzing the observed circuit behavior and identifying a failing region through methods like path tracing [6]. Next, different types of faults within the found failing region are simulated. The simulated fault responses are then compared with the observed response measured by the tester to determine candidates that best represent the defect. Each candidate may then also be relatively validated against its physical likelihood of actually corresponding to a defect based on its layout-level properties. Lastly, each candidate is ranked and assigned a score, which corresponds to the confidence the diagnosis algorithm has in

that candidate being correct.

Numerous approaches have been proposed over the years to improve the quality of diagnosis. The difference in these approaches primarily lies in the choice of fault models for simulation and the scoring procedure developed. For example, techniques such as [7]–[10] are based on logic fault models that assume an erroneous value of 0 or 1 at the fault location. However, not all defects (e.g., byzantine opens [11]), even for a subset of failing patterns, necessarily behave like a stuck-at fault. Thus, the X-fault model, where an unknown value (X) is assumed at the candidate location, is employed in [12]–[15] to avoid eliminating an actual defect location.

Various scoring methods exist in the literature to rank candidates. Methods described in [8]–[10], [16] are based on a logic fault model, but it might yield an incorrect set of candidates to begin with. On the other hand, methods discussed in [12]–[15] 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. More importantly, prior scoring methods rank candidates by a simple, fault-type independent expression that is created by intuition and domain knowledge. Conversely, a data driven scoring model (either dependent on or independent of a candidate's fault type) can discover (or learn) latent connections between the correct candidate and the tester response, and achieve better diagnosis accuracy and resolution.

It is thus not surprising that machine learning (ML) has been applied in the area of diagnosis. For instance, a random forest and a neural network are used in [17] and [18], respectively, to identify the defect type responsible for the observed failure. However, those approaches do not identify the defect location, which means they do not improve candidate-level resolution. In [19], a support vector machine is used on volume diagnosis data to improve the resolution of each individual diagnosis. Other applications of machine learning in volume diagnosis include finding failure-causing layout geometries using clustering [20], identifying Design-for-Manufacturing (DFM) rules whose violation caused a chip to fail using an expectation-maximization (EM) algorithm [21], and determining a failure root-cause distribution using an EM algorithm [3]. It should however be noted that the work yet-to-be-described can be used in conjunction with these ML-based volume diagnosis techniques to further improve resolution.

To address the shortcomings associated with different aspects of diagnosis discussed up to this point, this work 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 fault type (e.g., interconnect open, interconnect bridge and cell internal defect) and behavior. It should be noted that LearnX complements/strengthens approaches like [22], [23] where physical resolution is improved using layout neighborhood analysis.

2. Instead of just using a logic 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 since it allows an error to propagate from a defect location to the circuit outputs conservatively, which likely avoids removing a candidate that corresponds to an actual defect.

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 correlates to a defect. Specifically, a supervised learning algorithm is used to distinguish the correct candidate from incorrect ones.

As mentioned earlier, 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. The way this is enhanced by LearnX is described in the remaining sections of this manuscript. Specifically, Section II provides a detailed overview of various phases of LearnX. Its effectiveness is demonstrated via several experiments that are described in Section III. Finally, Section IV draws conclusions and provides several directions for future work.

II. Diagnosis Methodology

Fig. 1 shows the overview of LearnX. LearnX is a two-phase diagnosis methodology. The first phase (detailed in Section II-A) aims to identify defects that mimic 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 II-B). Such defects are identified 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 [12], 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.

A. *LearnX: Phase* 1

The first step in Phase 1 of LearnX is path tracing [6]. 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 [22]. Next, for each failing pattern, a stuck-at fault at each candidate location is simulated to find faults that

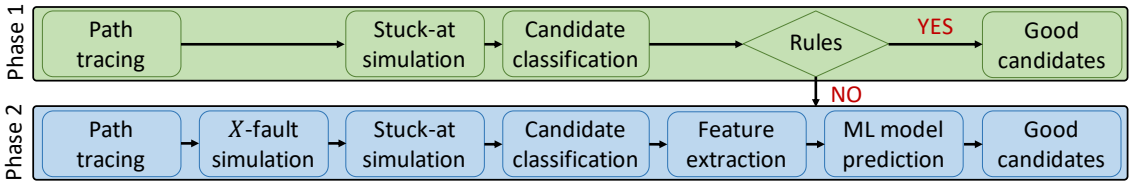

Fig. 1. Overview of the proposed diagnosis methodology, LearnX.

explain[1] that pattern [8]. 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 cover is classified into one of the following four fault types according to its behavior. A set of rules that are constructed for each fault type to identify the correct candidate are also explained next.

1. STUCK, when a cover contains only one fault. If the fault simulation response is identical to the tester response, the candidate is deemed correct.

2. CELL, when a cell candidate is consistent [24], [25]. That is, sets of logic values established on the cell inputs for Tester-Fail-Simulation-Fail (TFSF) and Tester-Pass-Simulation-Fail (TPSF) patterns are disjoint. Here, a TFSF (TPSF) is a pattern that fails (passes) on the tester and detects a stuck-at fault at the candidate location. Any cell candidate that is found consistent is adjudged correct. (Each consistent cell can further be inspected to derive intra-cell candidates [23].)

3. BRIDGE, when a cover consists of faults corresponding to physically adjacent nets. The candidate is deemed correct if the bridged nets have opposite polarities for each TFSF pattern and same polarities for each TPSF pattern.

4. OPEN, when a cover contains faults affecting the same signal. In addition to explaining each failing pattern, the candidate is deemed correct if at least one of the cover's constituent faults passes (but is excited) for each passing pattern.

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.

B. *LearnX: 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 [12]. 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 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. Next, a set of features for each candidate is extracted using test and manufacturing domain knowledge (Table 1). 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 by the tester 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[2], and are thus believed to capture a more complete picture. In addition, the features extracted here are more detailed. For example, the number of TPSF outputs are counted separately for TFSF and TPSF patterns (rows 5 and 8 in Table 1, respectively), instead of recording the total number of TPSF outputs over patterns that fail during simulation.

The next step in Phase 2 is to classify whether a candidate is correct or not using machine learning. A commonly used machine learning method called a random forest [26] 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. Each random forest is trained using virtual test responses generated through fault injection and simulation. Hyperparameters of each trained model are tuned using a separate validation data set.

Note that the training and the validation data sets are 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) is not optimum. Thus, a Precision-Recall (PR) curve [27] is used to find an optimum threshold. A PR curve shows the trade-off between precision, which is the proportion of the examples that are truly positive among the ones classified as positive, and recall, which is defined as the ratio of positive examples that are correctly classified, for different thresholds. Fig. 2 illustrates the effect of increasing threshold on precision and recall. In this work, the optimum

[1]A stuck-at fault is said to 'explain' a pattern if the circuit response predicted during its simulation is identical to the tester response.

[2]Twenty-two features listed in Table 1 are extracted for each type of simulation, resulting in a total of 44 features.

TABLE 1
FEATURES EXTRACTED FOR THE MACHINE LEARNING MODEL USED IN PHASE 2.

Feature	Description
$TFSF_p/TF_p$ ($TFSF_p/SF_p$)	The number of TFSF patterns divided by the number of TF (SF) patterns.
$TPSF_p/TP_p$ ($TPSF_p/TP_p$)	The number of TPSF patterns divided by the number of TP (SF) patterns.
$TFSP_p/TF_p$ ($TFSP_p/TF_p$)	The number of TFSP patterns divided by the number of TF (SP) patterns.
$TFSF_o\_TFSF_p/TF_o$ ($TFSF_o\_TFSF_p/SF_o$)	The number of TFSF outputs in TFSF patterns divided by the number of TF (SF) outputs.
$TPSF_o\_TFSF_p/TP_o$ ($TPSF_o\_TFSF_p/SF_o$)	The number of TPSF outputs in TFSF patterns divided by the number of TP (SF) outputs.
$TFSP_o\_TFSF_p/TF_o$ ($TFSP_o\_TFSF_p/SP_o$)	The number of TFSP outputs in TFSF patterns divided by the number of TF (SP) outputs.
$TPSP_o\_TFSF_p/TP_o$ ($TPSP_o\_TFSF_p/SP_o$)	The number of TPSP outputs in TFSF patterns divided by the number of TP (SP) outputs.
$TPSF_o\_TPSF_p/TP_o$ ($TPSF_o\_TPSF_p/SF_o$)	The number of TPSF outputs in TPSF patterns divided by the number of TP (SF) outputs.
$TPSP_o\_TPSF_p/TP_o$ ($TPSP_o\_TPSF_p/SP_o$)	The number of TPSP outputs in TPSF patterns divided by the number of TP (SP) outputs.
$TFSP_o\_TFSP_p/TF_o$ ($TFSP_o\_TFSP_p/SP_o$)	The number of TFSP outputs in TFSP patterns divided by the number of TF (SP) outputs.
$TPSP_o\_TFSP_p/TP_o$ ($TPSP_o\_TFSP_p/SP_o$)	The number of TPSP outputs in TFSP patterns divided by the number of TP (SP) outputs.

TF: Tester-Fail; TP: Tester-Pass; SF: Simulation-Fail; SP: Simulation-Pass
TFSF: Tester-Fail-Simulation-Fail; TPSF: Tester-Pass-Simulation-Fail; TFSP: Tester-Fail-Simulation-Pass; TPSP: Tester-Pass-Simulation-Pass

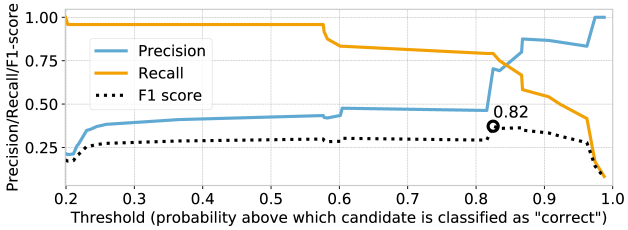

Fig. 2. Selecting the optimum decision threshold that maximizes the harmonic average between precision and recall (known as the F1-score [27]).

threshold for each trained forest is found by maximizing the *F1-score* (plotted with a black dotted line in Fig. 2), which is defined as the harmonic mean of precision and recall [27].

Each trained model inherently acts like a scoring framework, and assigns a probability (or a "score") to each defect candidate. 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 would differentiate correct candidates from incorrect ones.

III. EXPERIMENT

Two experiments are described to validate 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 using three designs is performed – one is a sub-circuit of the L2 cache write-back buffer (called L2B) of the OpenSPARC T2 processor [28], the second design is a logic characterization vehicle (LCV) [2] and the third design is an ITC99 benchmark circuit called B15, which is a subset of the Intel 80386 microprocessor [29]. For each design, the following fault types are considered to model realistic defect behaviors.

1. **Two-line bridge defects** are modeled using a variety of fault models such as the wired-bridge model (AND-type, OR-type and the dominating bridge) and the biased voting model (which includes the byzantine bridges, where error can manifest from both the nets for a pattern) [30]. Potential bridge net pairs are extracted from the design layout [22].
2. **Open defects** are 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 defect location for each sensitizing pattern [11]. Open defect locations are extracted from the layout [22].
3. **Cell defects**: Open, bridge and transistor defects are injected into the layout of each cell. Each resulting cell-level defect response is then simulated at the logic level to produce a "fail log".

For each design, a total of 15,000 virtual fail logs are created, of which 3,000 are used to create the training dataset and another 2,000 are used to generate the validation dataset. Thus, 10,000 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 two criteria - diagnostic resolution and accuracy. Resolution is defined as the number of candidate covers returned by each approach. (It should be noted that only the top-scoring candidates are considered for commercial diagnosis.) A defect is said to be diagnosed accurately if the location of the injected fault matches with a candidate. For a bridge fault, the degree of accuracy (whether one or both the bridged nets are reported) is also noted. An ideal diagnosis result would thus have resolution equal to one and 100% diagnostic accuracy.

Figures 3 and 4 show histograms of the resolution achieved by Phase 1 and Phase 2, respectively, from diagnosing virtual fail logs that correspond to the L2B design. The x-axis shows the resolution and the y-axis shows the number of fail logs. The percentage of fail logs accurately (i.e., 100% accuracy) 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 accurately diagnosed while the bottom half shows the distribution of the inaccurately diagnosed fail logs. The

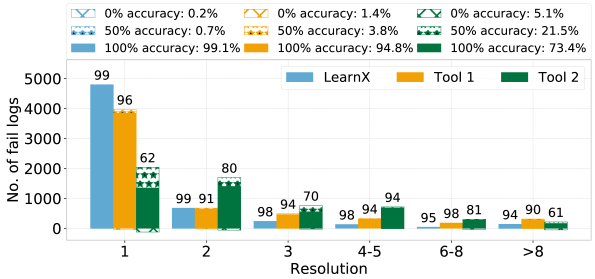

Fig. 3. Resolution distribution attained by Phase 1 for 6,035 virtual fail logs.

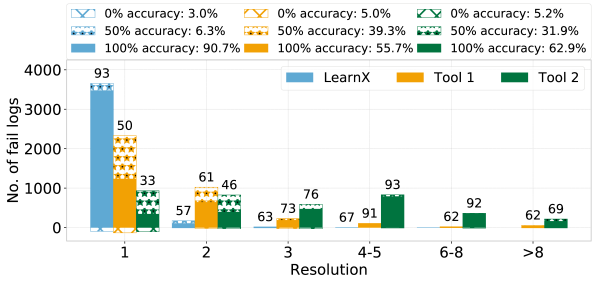

Fig. 4. Resolution distribution attained by Phase 2 for 3,965 virtual fail logs.

percentage of fail logs diagnosed accurately by each diagnosis technique is shown above the plot in each figure.

Fig. 3 reveals that LearnX achieves 100% accuracy for 99.1% fail logs, which is 4.5% (35%) more than Tool 1 (Tool 2). In addition, 79.6% of fail logs attain perfect resolution, an improvement of 19.7% over Tool 1 and 2.2X times Tool 2.

It is observed from Fig. 4 that LearnX attains 100% accuracy for 90.7% fail logs, which is 62.8% and 44.2% more than Tool 1 and Tool 2, respectively. In addition, 94.6% fail logs achieve a resolution of one, an enhancement of 52.1% over Tool 1 and 3.6X times Tool 2. Among the fail logs diagnosed with perfect resolution, LearnX diagnoses 92.6% fail logs with ideal accuracy, while Tool 1 and Tool 2 correctly diagnose only 50% and 32.6% of fail logs, respectively.

A summary of the improvement in diagnostic resolution and accuracy attained by LearnX over commercial diagnosis for three designs is shown in Table 2. Table 2 clearly shows that LearnX performs significantly better. On average, LearnX diagnoses 27.2% (64%) more fail logs with ideal accuracy and 39.0% (74%) more fail logs with ideal resolution when compared with Tool 1 (Tool 2). More importantly, LearnX returns an ideal diagnosis outcome (i.e., when a single candidate returned is correct) for 86.6% (2.2X) more fail logs than Tool 1 (Tool 2), on average.

B. Silicon

LearnX is evaluated on silicon failure data that is obtained for a test chip fabricated in a leading technology node. Unlike the simulation-based experiment described in Section III-A, the ground truth (i.e., the location and the nature of a defect) is not available, and hence only diagnostic resolution is compared between LearnX and commercial diagnosis.

Figures 5 and 6 show the resolution distribution for Phase 1 and Phase 2, respectively. It is seen from Fig. 5 that LearnX achieves perfect resolution for 53.4% fail logs, which is 0.7% and 4.4% less than Tool 1 and Tool 2, respectively. However,

TABLE 2
DIAGNOSTIC RESOLUTION AND ACCURACY IMPROVEMENT OVER TWO COMMERCIAL DIAGNOSIS TOOLS FOR DIFFERENT DESIGNS.

Design	100% accuracy (%)		Resolution = 1 (%)		Resolution = 1 & 100% accuracy (%)	
	Tool 1	Tool 2	Tool 1	Tool 2	Tool 1	Tool 2
L2B	20.8	38.4	32.1	167.1	62.4	388.0
LCV	42.0	98.7	37.8	-31.1	115.9	44.3
B15	18.9	55.0	47.0	86.1	81.6	233.8
Average	27.2	64.0	39.0	74.0	86.6	222.0

Note: For the LCV design, although LearnX returns a single candidate for 31.1% fewer fail logs when compared to Tool 2, it finds a single correct candidate for 56.6% fail logs, which is 44.3% more than Tool 2. In other words, the likelihood of a candidate being correct when a single candidate is returned is 46.3% when Tool 2 is used, and is 97.0% when LearnX is used.

it can be deduced from Table 2 that only 72.9% (51.2%) of the injected faults that are diagnosed with perfect resolution are accurately diagnosed when Tool 1 (Tool 2) is used. On the other hand, LearnX returns a single candidate accurately for 96.9% of the injected faults. Thus, even if it may appear from Fig. 5 that LearnX attains a slightly lower resolution than a commercial tool, LearnX would yield a candidate set that includes the correct candidate more often.

It is observed from Fig. 6 that Phase 2 attains a resolution of one for 70.7% fail logs, an improvement of 17.4% and 20.7% over Tool 1 and Tool 2, respectively. Although the accuracy cannot be compared here due to the lack of PFA results, it can be extrapolated from Fig. 4 and Table 2 that Phase 2 would find the correct candidate significantly more often than any of the two commercial diagnosis tools.

The improved performance of LearnX over commercial tools comes at the expense of 12% increase in diagnosis runtime. In addition, there is a one-time cost involved in training the ML model, which is up to two hours in the experiments we conducted. However, as evident from Figures 3-6 and Table 2, the additional steps involved in LearnX (i.e., X-fault simulation, and ML training and inference) enhance the quality of diagnosis (in terms of diagnosis resolution and accuracy) significantly, which consequently, would improve the effectiveness of PFA, and likely facilitate yield learning.

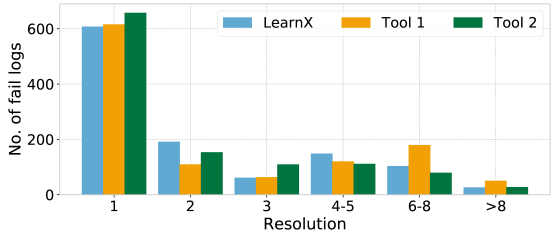

Fig. 5. Resolution distribution attained by Phase 1 for 1,136 silicon fail logs.

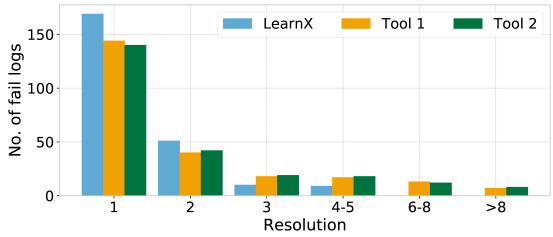

Fig. 6. Resolution distribution attained by Phase 2 for 239 silicon fail logs.

IV. Conclusions

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 outputs.

Several experiments are conducted to evaluate the performance of LearnX. Simulation-based experiments carried out for three different designs with 10,000 faulty circuits each (that are created using a variety of realistic defect behaviors) reveal that LearnX achieves a resolution of one for 75.6% of the injected faults, showing an improvement of at least 39.0% over state-of-the-art commercial diagnosis. Additionally, the average ideal accuracy of LearnX is 95.7%, which is 27.2% and 64.0% higher than two commercial diagnosis tools utilized here. Moreover, LearnX returns an ideal diagnosis outcome (i.e., a single candidate that correctly represents the injected fault) for 73.2% of the injected faults, an improvement of 86.6% over the better-of-the-two commercial diagnosis tools.

Significance of LearnX is further substantiated by diagnosing 1,375 silicon fail logs from a design fabricated in an advanced process technology. LearnX and commercial diagnosis appear to produce a similar resolution histogram; however, given the superior diagnosis quality attained by LearnX in the aforementioned simulation experiments, it can be said with confidence that LearnX would identify the correct candidate for more fail logs than commercial diagnosis when PFA results are available for these failures.

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. Future work focuses on extending LearnX to incorporate sequence- and timing-dependent defects, and extracting design-specific features [17], [19] in the second phase to further improve its performance.

References

[1] R. Desineni, Z. Berndlmaier, J. Winslow, A. Blauberg, and B. R. Chu, "The grand pareto: A methodology for identifying and quantifying yield detractors in volume semiconductor manufacturing," *IEEE Transactions on Semiconductor Manufacturing*, vol. 20, no. 2, pp. 87–100, May 2007.

[2] B. Niewenhuis, "A logic test chip for optimal test and diagnosis," Ph.D. dissertation, Carnegie Mellon University, 2018.

[3] 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.

[4] 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.

[5] Y. T. Lin and R. D. Blanton, "METER: Measuring test effectiveness regionally," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 7, pp. 1058–1071, July 2011.

[6] 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.

[7] S. Venkataraman and S. B. Drummonds, "POIROT: A logic fault diagnosis tool and its applications," in *IEEE International Test Conference*, 2000.

[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*, 2001, pp. 287–296.

[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] S.-Y. Huang, "Diagnosis of byzantine open-segment faults," in *IEEE Asian Test Symposium*, Nov 2002, pp. 248–253.

[12] 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.

[13] 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*, Nov 2004.

[14] 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.

[15] X. Wen, H. Tamamoto, K. K. Saluja, and K. Kinoshita, "Fault diagnosis for physical defects of unknown behaviors," in *IEEE Asian Test Symposium*, Nov 2003.

[16] I. Pomeranz, "OBO: An output-by-output scoring algorithm for fault diagnosis," in *IEEE Computer Society Annual Symposium on VLSI*, 2014.

[17] J. E. Nelson, W. C. Tam, and R. D. Blanton, "Automatic classification of bridge defects," in *IEEE International Test Conference*, Nov 2010.

[18] L. R. Gómez and H. Wunderlich, "A neural-network-based fault classifier," in *IEEE Asian Test Symposium*, Nov 2016.

[19] 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*, vol. 37, no. 6, pp. 1288–1297, June 2018.

[20] 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*, vol. 34, no. 8, pp. 1278–1290, Aug 2015.

[21] 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*, vol. 36, no. 3, pp. 463–474, March 2017.

[22] S. Mittal and R. D. Blanton, "PADLOC: Physically-aware defect localization and characterization," in *IEEE Asian Test Symposium*, Nov 2017.

[23] ——, "NOIDA: Noise-resistant intra-cell diagnosis," in *IEEE VLSI Test Symposium*, April 2018.

[24] M. E. Amyeen, D. Nayak, and S. Venkataraman, "Improving precision using mixed-level fault diagnosis," in *IEEE International Test Conference*, Oct 2006.

[25] 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.

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

[27] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[28] I. Parulkar *et al.*, "OpenSPARC: An open platform for hardware reliability experimentation," in *Workshop on Silicon Errors in Logic-System Effects*, 2008.

[29] 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.

[30] 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.