Adaptive Test Pattern Reordering for Diagnosis using k-Nearest Neighbors

Chenlei Fang, Qicheng Huang and R. D. (Shawn) Blanton
Advanced Chip Testing Laboratory (www.ece.cmu.edu/~actl/)
Department of Electrical and Computer Engineering
Carnegie Mellon University, Pittsburgh, PA 15213
{chenleif, qichengh, rblanton}@andrew.cmu.edu

***Abstract*—Logic diagnosis is a software-based methodology to identify the behavior and location of defects in failing integrated circuits, which is an essential step in yield learning. However, accurate diagnosis requires a sufficient amount of failing data, which is in contradiction to the requirement of reducing test time and cost. In this work, a dynamic test pattern reordering method is proposed to "recommend" which test patterns should be applied for a given failing chip, with the goal of maximizing failing data while minimizing test time. Unlike prior work that uses population statistics from already tested chips, this method uses a machine learning technique, namely k-Nearest Neighbors. Experiments using three industrial chips demonstrate the efficacy of the proposed methodology; specifically, the recommended test pattern order led to a 35% reduction, on average, while maximizing the amount of failure data collected.**

I. Introduction

Diagnosis of failing integrated circuits (IC) is a key step during yield ramping and high-volume manufacturing. It is a software-based process which takes place after an IC fails testing, with the goal of identifying possible locations and/or behavior of defects within a failing chip. The inputs for diagnosis of a scan-based failure includes (i) design descriptions (logical netlist and often a layout), (ii) test patterns, and (iii) tester responses (also referred to as the fail log). The result of diagnosis provides useful information for further analysis. For example, diagnosis reports likely defect locations for physical failure analysis (PFA) which is a process for understanding the exact cause of the failure. Volume diagnosis techniques, which is the statistical analysis of diagnosis results from a large population of chip fails, is useful understanding defect distribution, defect systematics, etc. [1-4]. In industry, diagnosis is one of many fabrication monitors used during yield ramping and high-volume manufacturing.

The quality of a diagnosis is generally measured in terms of its resolution and accuracy. Resolution refers to the number of possible defect locations, and a diagnosis is deemed accurate if the failing location is among the sites reported. Two different approaches exist for performing diagnosis: cause-effect and effect-cause. Cause-effect method simulates all possible faults (typically single stuck-at faults), stores the faulty outputs as a fault dictionary, and compares the outputs to circuit under test (CUT) responses. Effect-cause, on the other hand, starts from "back-coning" from the erromeous outputs and identifies the candidate faulty locations, as shown in Fig. **1**. Most state-of-the-art diagnosis tools adopt a combination of these two approaches [5]. That is, the tool first uses effect-cause to find a rather small portion of the CUT, then simulates faults in the portion in a cause-effect manner.

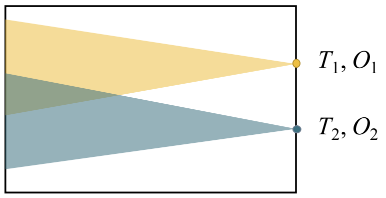

Fig. 1. An example of back-coning a CUT with two failing patterns; T_1 with output O_1 failing, and T_2 with output O_2 failing.

In general, the more tester-response collected, a more precise diagnosis can be obtained. For example, as shown in Fig. **1**, suppose we have a chip failing for pattern T_1 at output O_1, and T_2 at output O_2. If only the tester response for pattern T_1 is at hand, then the entire back-cone of O_1 will be possible candidates for the effect-cause step. Even if fault simulation is executed for all these locations to rule out some candidates, many candidates will still remain, resulting in poor resolution. However, if the responses from pattern T_2 is also included for diagnosis, the possible candidate area can shrink to the overlap of the two back-cones, leading to a potentially better resolution (assuming only one defect exists). The need to collect sufficient fail data is not only stated in the user manual of commercially-available tools[1], and is often verified in practice. For example, for a randomly-selected set of 69 chips that are diagnosed to have single defect, we truncate each corresponding fail log to observe the impact of reduced fail data. The first trial uses data from 30 failing patterns, while the second uses data from five failing patterns. Comparing the diagnosis outcomes from the two trials reveals that resolution is degraded for more than 50% of the chips.

Despite the benefit brought by additional fail data, collecting more data is not always feasible. For example, collecting more tester data increases cost due to increased tester time, and may even increase tester cost if more memory is needed to store fail data.

There are two categories of prior work that attempt to mitigate the trade-off between test costs and diagnosis quality.

[1]The user manual for Tessent, a diagnosis tool from Mentor Graphics, states the following "Include failures from at least 30 failing scan test patterns to achieve good logic diagnosis resolution. Mentor Graphics recommends 100 failing scan test patterns for optimal results." [6].

One category focuses on adaptive testing to reduce test cost. For example, [7] suggests reordering the test patterns such that the patterns are sorted based on their capability to detect fails on chips already tested. The most effective patterns are applied first so that bad chips are discovered quickly. The authors of [8] describe a wafer-level pattern sampling approach to identify and apply a subset of the tests on a subset of the wafers for reducing test cost. These works all focus on "detecting" failing parts, but are not diagnosis-oriented. This work, and others like it, all aim to minimize test time, and thus are not focused on ensuring sufficient fail data for diagnosis.

The other category of prior work improves diagnosis quality by manipulating the test patterns applied, or adaptively changing the fail-data collection process. The work in [9, 10] assume that the test patterns cannot be changed, but the fail data to be recorded can be chosen to optimize diagnosis outcomes. This approach is intended for situations where ATE memory is limited. The work in [11] trains a machine learning (ML) model to decide when enough fail data is collected for an accurate diagnosis. The approaches described in [12, 13] statically reorder test patterns for distinguishing faults within the window of fail-data collection.

In this paper, an adaptive test pattern reordering algorithm is described, which collects sufficient fail data to ensure good diagnosis outcomes with fewer applied test patterns. The approach used here is based on the personalized movie/book recommending systems that are widely adopted. Here, failing test patterns of already-tested chips that are similar to the current CUT are recommended for application. It is more likely that the current chip will also fail for these recommended test patterns. The k-Nearest Neighbors (kNN) algorithm is implemented to accomplish this task. Unlike prior work, the approach described here does not require extensive computation for fault simulation and model training. In addition, it directly accounts for the characteristics of the relevant defects because it utilizes historical data from the same design/fabrication environment for making test-pattern recommendations.

The rest of this paper is organized as follows. Section II describes the formulation of the problem and details of the pattern reordering methodology. Section III demonstrates the efficacy of our method in reordering the test patterns for two types of industrial chips. Finally, Section IV concludes the paper and provides directions for future work.

II. Methodology

A pattern reordering flow is developed in this work to achieve the goal of collecting sufficient fail data with reduced test cost. As described in Section I, this problem is similar to the recommendation systems, which can be built using kNN. First, an overview of common recommendation systems is given, followed by a discussion on altering recommendation for adaptive testing in Section II-A. The pattern reordering algorithm using kNN is described in detail in II-B. Finally, important implementation details, including the distance metrics and online learning, are described in II-C.

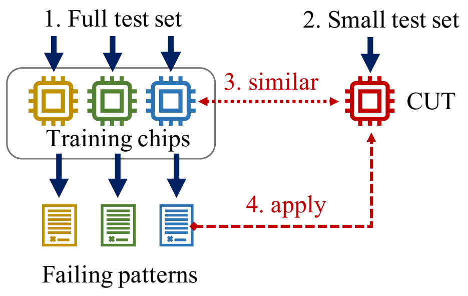

Fig. 2. Pattern reordering based on chip similarity.

A. *Overview*

Movie/book/product recommending systems are widely used in daily life. These systems usually build a "user-item" model that is based on the following assumption: the preference of one individual can be predicted based on preferences of other similar individuals. Once a user creates an account, and indicates several preferences, the system will automatically recommend items that are popular with similar users.

In our application, the goal is to dynamically rearrange a fixed set of test patterns to produce an optimal diagnosis outcome on a per chip basis. That is, each chip, we want to apply the patterns that the chip is more likely to fail before other patterns that are more likely to pass. A common property found in semiconductor fabrication is exploited to achieve the goal: defects occurring for a given design and fabrication process will not be completely random, but instead also include so-called systematic defects that are similar in location and behavior. This property implies that chips with similar defects will fail for similar test patterns. In this way, the chips and test patterns act as the "users" and "items", respectively, in a recommending system. The similarity between these techniques motivates the use of recommending system approaches for test reordering.

A flow diagram for pattern reordering is shown in Fig. **2**. In the first step, a set of failing chips that have been fully tested are required for the training set. Next, a CUT has a small set of test applied to obtain the data for making a recommendation of the next test pattern. Specifically, the pattern reordering algorithm will use the data to recommend test patterns based on similar failing chips found within the training set. The failing patterns of these similar chips are applied to the CUT. The details of the reordering algorithm are discussed in next subsection.

B. *Pattern Reordering*

Recommending systems typically measure similarity using the well-known kNN algorithm. kNN makes no assumptions on the underlying data distribution, and does not require a training process typical of other machine learning algorithms, because there is no model to build. kNN is based merely on comparing the feature similarity between existing and new data instances.

In this work, a failing test pattern signature (i.e., the number of failing outputs for each test pattern) is adopted as the feature

	O_1	O_2	...	...	O_m	Pattern Signature
T_1		×				1
T_2		×		×	×	3
...	×					1
...						0
T_n			×			1

Fig. 3. Illustration of fail-log signatures, where × stands for a failure at the corresponding output and test pattern.

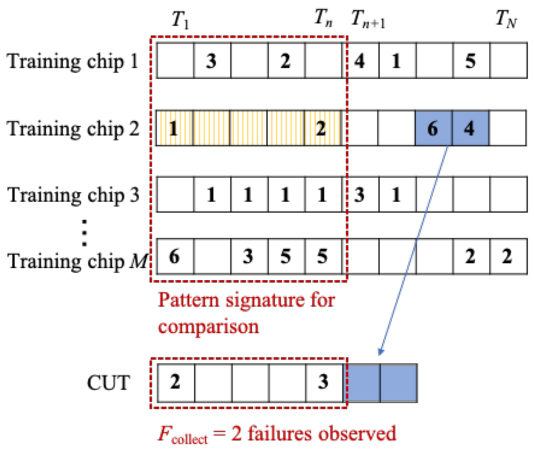

Fig. 4. A pattern reordering example using kNN.

for similarity comparison. Fig. **3** shows how to construct a pattern signature for one chip, where the table illustrates a typical fail log. $T_1 \sim T_n$ represent the n patterns applied, and the failing outputs are recorded and marked with an "×". The number of failing outputs for each test pattern is counted and recorded as the *pattern signature*.

An example of the kNN based pattern reordering algorithm using a pattern signature is depicted in Fig. **4**. We assume M failing chips have already been tested with the full test set T with N patterns. The corresponding pattern signatures derived from the N failing chips serve as training data as a matrix of size $M \times N$. When a new chip is being tested, for similarity comparison, the new CUT is tested with a limited number of patterns from T until F_{collect} failing patterns are observed. In this example, $F_{\text{collect}} = 2$, and $n = 5$ patterns are applied so far. The number of failing patterns required for comparison F_{collect} is selected arbitrarily, but the number of patterns applied n can differ for each CUT. The pattern signature vector with length $n << N$ is used as the feature vector for computation of similarity. In the kNN algorithm, the k training chips whose pattern signatures from patterns 1 to n are closest to the CUT are identified. For simplicity, $k = 1$ is used in this example, and training chip 2 is the nearest neighbor (yellow) to the CUT, that is, their signatures are very similar. (The distance metric is discussed in the next subsection.) Once the k nearest signatures are identified, the additional failing patterns for these k signatures are derived, by calculating the largest of the pattern signatures from entries $n+1$ to N. The most frequent failing patterns (shaded in blue) are applied to the CUT. Once a sufficient number of failing patterns are observed, testing is terminated.

C. Implementation

When implementing the kNN algorithm, the definition of "similarity" is very critical. Because 1-D feature vectors are used, there are several approaches available to calculate the similarity measure. The most commonly used approach is cosine similarity, which is defined as:

$$m_1 = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\Sigma_i A_i B_i}{\sqrt{\Sigma_i A_i^2}\sqrt{\Sigma_i B_i^2}}, \tag{1}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the two feature vectors to be compared.

A second approach uses hamming distance. Given that the goal is to observe more failing patterns, what is more important is whether each pattern fails or not, while the number of failing outputs for each pattern is less significant. This observation motivates the use of a binarized pattern signature, where hamming distance is used to measure similarity. The hamming distance between two signatures $\mathbf{A}$ and $\mathbf{B}$ is defined as:

$$\begin{aligned} \mathbf{A}' &= \mathbb{1}(\mathbf{A} > 0) \\ \mathbf{B}' &= \mathbb{1}(\mathbf{B} > 0) \\ m_2 &= |\mathbf{A}' - \mathbf{B}'| = \Sigma_i |A_i' - B_i'|, \end{aligned} \tag{2}$$

where $\mathbb{1}(\cdot)$ is the indicator function, which equals to 1 when the condition is satisfied.

A third possible measure involves combining Eq. (1) and Eq. (2), that is, calculating the cosine similarity of two binarized signatures:

$$\begin{aligned} \mathbf{A}' &= \mathbb{1}(\mathbf{A} > 0) \\ \mathbf{B}' &= \mathbb{1}(\mathbf{B} > 0) \\ m_3 &= \frac{\mathbf{A}' \cdot \mathbf{B}'}{\|\mathbf{A}'\|\|\mathbf{B}'\|}. \end{aligned} \tag{3}$$

Cross validation can be used to choose the best similarity measure. In experiments, it is observed that the different measures lead to slightly different performances.

Choosing the value of k is also very important. A small value for k is susceptible to random noise, while a large k is less capable of capturing complex defect behavior. If the value k is chosen to be the number of training chips, it simply means the most frequent failing patterns will be recommended for each CUT. Such a strategy is not adaptive, and is later shown to be ineffective when compared to adaptive methods. The best value for k can also be found via cross validation.

Another strategy deployed to enhance performance is online learning. At the beginning of the flow in Fig. **2**, only a small set of chips, e.g., 20% of the total chips are fully tested to serve as the training data. This training data may not be sufficient, as a majority of the chips remain unseen. A better strategy is to continuously update the training set as chips are tested. Unlike other ML methods, kNN easily adapts to online learning, because it does not have an explicit training stage.

In light of the aforementioned discussion, the pattern reordering procedure with online learning is describe in Algorithm 1:

```
Algorithm 1 Online Learning for Pattern Reordering
Initialization: $\mathbf{X}_{\text{train}}$ = fail-log signatures for a set of fully-test chips
while for each CUT do
    1. Test CUT until $F_{\text{collect}}$ failing patterns occur
    2. Identify $k$ nearest signatures $X_1, ..., X_k$ with length $n$ from $\mathbf{X}_{\text{train}}$
    3. Sort $n+1, ..., N$ failing patterns of the $k$ signatures from most to the least frequent
    4. Apply the ordered patterns until the limit
    5. Construct signature $X_{\text{CUT}}$
    6. $\mathbf{X}_{\text{train}} \leftarrow \mathbf{X}_{\text{train}} \cup X_{\text{CUT}}$
end
```

III. Experiments

In this section, experiment details based on three industrial chips are discussed to demonstrate the efficacy of adaptive test reordering for diagnosis.

A. Setup

Because diagnosis is an essential step for process development and high-volume manufacturing, two types of industrial chips are used in the experiments. Chip 1 is a high-volume design manufactured in a 90nm process. Chip 2 is a logic test chip used in development of a 28nm fabrication process. Chip 3 is another test chip manufactured in a 14nm technology. Their characteristics are listed in Table I. Industrial partners executed circuit tests on manufactured chips, and fail logs from these three chips are used for the forthcoming experiments.

B. Comparison Measures

To demonstrate the efficacy of the developed method, the test patterns needed of the adaptive approach is compared with the default test pattern order. The experiment setting is the same as discussed in Section II-B, that is, M chips are fully tested, and their pattern signatures are extracted as training samples. For a new CUT, F_{collect} patterns are collected for kNN comparison. In the following experiments, the initial train/test split ratio is 20/80, that is, 20% of total chips are used as training samples. Because online training is adopted, the number of training samples increase as more chips are tested. The number of failing patterns assumed for an accurate diagnosis, F_{obj}, is varied in the experiments.

Assume M_{test} chips are tested with two orders: the rearranged order suggested by the adaptive method, and the default order. For chip i, define N_i^{reorder} to be the number of patterns applied until $F_{\text{obj}} - F_{\text{collect}}$ more failing patterns are observed using the reordered patterns (so that F_{obj} failing

TABLE I
CHARACTERISTICS OF TWO INDUSTRIAL CHIPS.

Chip Identifier	Chip 1	Chip 2	Chip 3
Chip type	High volume chip	Test chip	Test chip
Technology	90nm	28nm	14nm
No. of standard cells	9.3 M	4.4 M	4.2 M
No. of scan chains	103	12	12
Test set size	1,000	500	824
No. of fail logs	9,301	4,235	570

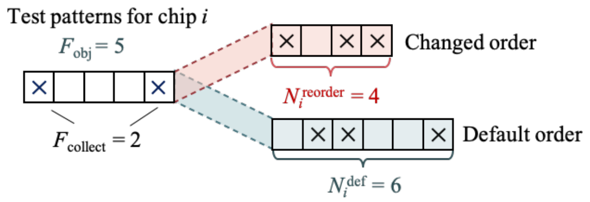

Fig. 5. An example shows the definitions of F_{obj}, F_{collect}, N_i^{def} and N_i^{reorder} for test pattern comparison.

patterns are collected in total), and N_i^{def} is the number of patterns applied with the default order to observe the same number of failing patterns. We assume test terminates when the limit F_{obj} is satisfied, so N_i^{def} and N_i^{reorder} represent test cost for the CUT with the two orders, respectively. An illustration of these variables is given in Fig. **5**. Test pattern reduction is calculated using the three following measures:

- Win percentage of the reordered patterns:
$$\text{Win\%} = \frac{\Sigma_{i=1}^{M_{\text{test}}} \mathbb{1}(N_i^{\text{reorder}} < N_i^{\text{def}})}{M_{\text{test}}} \times 100\%, \tag{4}$$
where reorder "wins" if the approach collects F_{obj} failing patterns with fewer applied patterns than the default ordering. Eq. (4) counts how often the adaptively reordered patterns is beneficial for test pattern reduction.
- The reduction in the number of patterns applied by reordering is given by:
$$\text{Reduction count} = \Sigma_{i=1}^{M_{\text{test}}} N_i^{\text{def}} - N_i^{\text{reorder}}. \tag{5}$$
Unlike win%, this measure determines the magnitude of test cost savings via pattern count, which is reasonable because cost is proportional to the number of applied test patterns.
- The average percentage of patterns reduced per chip due to reordering is given by:
$$\text{Reduction\%} = \left(\Sigma_{i=1}^{M_{\text{test}}} \frac{N_i^{\text{def}} - N_i^{\text{reorder}}}{N_i^{\text{def}}}\right) \Big/ M_{\text{test}} \times 100\%. \tag{6}$$

C. Test Pattern Reduction

Various experiments are employed to determine the most effective similarity measure, and the value for the hyper-parameter k. The similarity metrics are analyzed by performing cross validation [14] on Chip 1 by randomly splitting the training/testing set five times. For each split, the pattern reordering algorithm is applied to the testing set, and the average performance of these executions is used to represent the performance of each similarity measure. The other parameters are chosen as follows: $F_{\text{obj}} = 50$, $F_{\text{collect}} = 10$, and $k = 20$. The results for the three similarity measures are shown in Table II. It can be observed that, among the three metrics, the binarized cosine similarity performs the best. A possible explanation is that, other than how many outputs fail for each pattern, it matters most whether a pattern fails or not, so binarizing the pattern signature makes sense here. Unlike hamming distance, cosine similarity is the inner product divided by the product of norms, which incorporates number of failing patterns for each

TABLE II
TEST PATTERN REDUCTION USING DIFFERENT SIMILARITY MEASURES.

	cosine	**hamming**	**binarized cosine**
Win%	92.5%	91.6%	**96.5%**
Reduction count	400,540	368,849	**442,073**
Reduction%	29.4%	30.9%	**35.2%**

TABLE III
TEST PATTERN REDUCTION FOR DIFFERENT VALUES OF k.

	k**=10**	k**=20**	k**=50**	k**=100**	k**=200**
Win%	93.7%	96.5%	97.3%	**97.5%**	97.3%
Reduction Count	406,000	442,073	**449,591**	444,496	432,858
Reduction%	32.1%	35.2%	**35.7%**	35.3%	34.6%

chip. As a result, the binarized cosine similarity is adopted for the following experiments.

Similarly, the choice of k for kNN can also be obtained using cross validation. Using the same procedure for choosing the similarity measure, cross validation experiments on Chip 1 are performed for values of k in the range $10 < k < 200$. The experiment results in Table III reveal that test pattern reduction first increases with a larger k, then decreases, and reaches the optimum when k is around 50. Although win% at $k = 100$ is slightly higher than that at $k = 50$, the other two performance measures achieve their largest value when $k = 50$. Therefore, $k = 50$ is selected for the follow-on experiments.

For the chosen similarity measure and hyper-parameter k, the efficiency of dynamic pattern reordering for Chip 1 is shown in Fig. **6**. Experiments are executed using varying values for F_{obj} and F_{collect}. The advantage of dynamic pattern reordering is more obvious when F_{obj} is larger, because there exists more space for improvement. When $F_{\text{obj}} > 25$, the probability that the reordered pattern "wins" exceeds 90%. The level of pattern reduction count is also significant. The greatest reduction count is > 500K patterns. Finally, the third measure, reduction%, is as high as 35% compared to the default order.

The same experiments are executed on Chip 2 as well, with results shown in Fig. **7**. Because Chip 2 is a test chip and the manufacturing process is not mature, more random defects, as well as complex defect mechanisms are likely to affect Chip 2. As a result, the similarity of failing chips is expected to be reduced. With that said, test pattern reduction is still observed for most values of F_{obj} and F_{collect}. Win% > 65%, and the applied test patterns are reduced by ~7,000.

In addition to the default order, a simple approach to pattern reordering is to apply the most frequently failing patterns first, as suggested in [7]. In this experiment, we also compare the test pattern reduction measures for dynamic pattern reordering and the static reordering method of [7]. The most frequent failing reordering is described in Algorithm 2.

TABLE IV
TEST PATTERN REDUCTION OF CHIP 3.

F_{collect}	**5**	**5**	**9**	**9**
F_{obj}	**15**	**45**	**45**	**65**
Win%	48.6%	48.0%	45.5%	43.6%
Reduction Count	262	168	164	-579
Reduction%	-1.8%	0.01%	-0.04%	-1.0%

Algorithm 2 Pattern Reordering using Frequency

```
Initialization: $\mathbf{X}_{\text{train}}$ = fail-log signatures for a set of fully-test chips
while for each CUT do
    1. Sort the failing patterns of $\mathbf{X}_{\text{train}}$ from the most frequent to the least frequent
    2. Apply the ordered patterns until the limit
    3. Construct signature $X_{\text{CUT}}$
    4. $\mathbf{X}_{\text{train}} \leftarrow \mathbf{X}_{\text{train}} \cup X_{\text{CUT}}$
end
```

The test pattern reduction measures for the two reordering approaches are plotted in Fig. **6** and Fig. **7**, where the red curve (labeled as "Most frequent") corresponds to the approach of [7]. Fig. **6** and Fig. **7** reveal that dynamic pattern reordering performs significantly better than the most frequent failing approach for all three test-cost reduction measures. This outcome demonstrates the advantage of customizing the pattern reordering over simply applying the most-frequently failing patterns first.

Despite the good performance on Chip 1 and 2, the pattern reordering method does not perform very well on Chip 3. The metrics calculated with several F_{obj} and F_{collect} are listed in Table. IV (randomly repeated five times). The win% is around 50%, the reduction count and reduction% are near zero, and sometimes even negative. A reason for this outcome is that Chip 3 is a test chip manufactured in a new technology where defect density is extremely high, likely resulting in failures caused by the presence of multiple defects. This explanation is evidenced by the large number of failing patterns observed for each failing chip. On average, each chip fails for 287 patterns, more than 1/3 of total (824) test set. In contrast, the number of failing patterns, on average, is 124 out of 1,000, and 56 out of 500 for Chips 1 and 2, respectively. A high failing rate suggests that a sufficient number of failing patterns can be easily collected with the default pattern order, which means the benefit of dynamic pattern reordering is negligible. Although the results suggest that dynamic pattern reordering does not provide additional benefits, the number of test patterns does not increase, thus demonstrating the robustness of this approach.

IV. CONCLUSION

In this work, an adaptive test pattern reordering method is described for reducing test cost while preserving diagnosis quality. The framework borrows the idea from the widely used recommendation systems to "recommend" patterns to each individual chip under test. Dynamic pattern reordering is based on the kNN algorithm, using pattern signatures from chop fail logs for similarity comparison. Experiments conducted

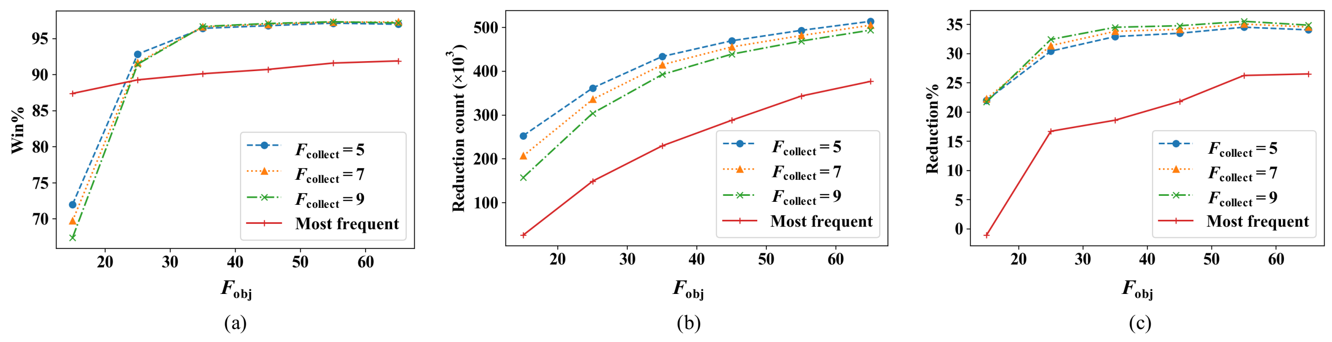

Figure 6. Test reduction results for Chip 1 for varying values of F_{obj}: (a) win%, (b) reduction count, and (c) reduction%.

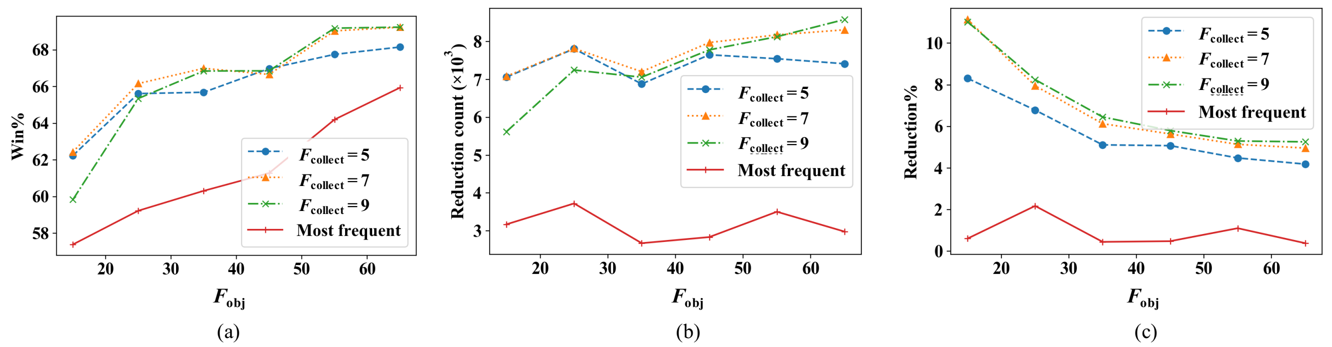

Figure 7. Test reduction results for Chip 2 for varying values of F_{obj}: (a) win%, (b) reduction count, and (c) reduction%.

on several fabricated chips have demonstrated that dynamic pattern reordering can collect sufficient failing patterns for high-quality diagnosis with significantly fewer test patterns compared to the default order or the frequency-based method of prior work [7]. For our experiments, it is assumed that the reordering algorithm is conducted online, which means the CUT does not have to be taken off the tester for offline computation. This assumption is reasonable because modern testers have the computation capability to perform ML inference, so the algorithm described in this paper is able to be conducted on the tester with proper settings. Therefore, the test patterns can be reordered dynamically on the tester, which means applying the algorithm does not introduce extra test cost.

V. Acknowledgement

The authors would like to thank their industrial collaborators for providing the diagnosis and failing data that enabled the silicon experiments in Section III.

References

[1] J. E. Nelson *et al.*, "Extracting Defect Density and Size Distributions from Product ICs," *Design & Test of Computers*, vol. 23, no. 5, pp. 390–400, 2006.

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

[3] B. Benware *et al.*, "Determining a Failure Root Cause Distribution from a Population of Layout-aware Scan Diagnosis Results," *Design & Test of Computers*, vol. 29, no. 1, pp. 8–18, 2012.

[4] R. D. Blanton *et al.*, "DFM Evaluation Using IC Diagnosis Data," *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 3, pp. 463–474, 2017.

[5] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital systems testing and testable design*. Computer science press New York, 1990, vol. 2.

[6] *Tessent Diagnosis User's Manual*. Mentor Graphics, 2019.

[7] R. Madge *et al.*, "In Search of the Optimum Test Set-Adaptive Test Methods for Maximum Defect Coverage and Lowest Test Cost," *International Test Conference*, pp. 203–212, 2004.

[8] M. Grady *et al.*, "Adaptive Testing-Cost Reduction through Test Pattern Sampling," *International Test Conference*, pp. 1–8, 2013.

[9] S. Tanwir *et al.*, "Information-theoretic and statistical methods of failure log selection for improved diagnosis," *International Test Conference*, pp. 1–10, 2015.

[10] S. Tanwir, M. Hsiao, and L. Lingappan, "A Test Pattern Quality Metric for Diagnosis of Multiple Stuck-at and Transition faults," *Great Lakes Symposium on VLSI*, pp. 455–458, 2017.

[11] H. Wang *et al.*, "Test-Data Volume Optimization for Diagnosis," *Design Automation Conference*, pp. 567–572, 2012.

[12] C. Xue and R. Blanton, "Test-Set Reordering for Improving Diagnosability," *VLSI Test Symposium*, pp. 1–6, 2017.

[13] G. Chen *et al.*, "A Test Pattern Ordering Algorithm for Diagnosis with Truncated Fail Data," *Design Automation Conference*, pp. 399–404, 2006.

[14] M. Stone, "Cross-validatory Choice and Assessment of Statistical Predictions," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 36, no. 2, pp. 111–133, 1974.