Chip test pattern reordering method using adaptive test to reduce cost for testing of ICs

Tai Song\(^1\), Huaguo Liang\(^1,\)\(^a\), Zhengfeng Huang\(^1\), Tianming Ni\(^2\), and Ying Sun\(^3\)

Abstract With the continuous drive toward integrated circuits (ICs) scaling, more test patterns are required in testing. However, the large number of patterns continues to increase test time and test costs. Thus, test costs of ICs are becoming more crucial yet more challenging. In this paper, we propose a novel adaptive test strategy to reduce test costs without increasing test escape, and using shortest path first (SPF) algorithm combined with K-Nearest Neighbor (KNN) to reorder the test patterns. The patterns which identify faults earlier are moved forward and applied first so as to save test time. In addition, the optimal test patterns are searched by means of a polynomial regression function and it provides a trade-off between test cost and test escape. Consequently, the optimization problem is converted into a mathematical function, in order to achieve broader applicability and generality. Experimental results demonstrate that the proposed method achieved 54.2 seconds time savings but leads to almost negligible test escapes increasing compared with traditional methods. The test pattern reordering method aims to find defects as early as possible. Furthermore, the proposed algorithm is completely software based and incurs no additional hardware overhead.

Keywords: adaptive test, machine learning, test pattern, test cost, pattern reordering

Classification: Integrated circuits (memory, logic, analog, RF, sensor)

1. Introduction

The continuous scaling of ICs technologies along with the increase in state-of-the-art design complexity have exacerbated the challenges associated with circuits testing. With such scaling, a large number of irrelevant test patterns and multiple test types with overlapping defect detection stand out among the most prominent factors limiting the test cost \cite{1}.

Traditionally, the test content, test order and test limits are fixed and only occasionally updated based on expert experience \cite{2}. However, such experience is easy to make mistake and an automated alternative is desired. Furthermore, traditional approach incurs very high test cost, since test patterns are required and performed one by one until it passes the whole suite of tests or fails one of the tests \cite{3}. This approach is satisfactory when circuit size is small, but when the circuit scale is larger, the traditional methods will be insufficient. This aggressive downscaling of dimensions in the forthcoming advanced technology generations poses critical testability issues \cite{4}, especially the test cost keeps increasing with the advancement of technology nodes.

Some irrelevant or redundant test patterns exist in the increased test set, and the effectiveness of the test pattern is reduced \cite{5}. New test types such as N-detect \cite{6} and gate-exhaustive tests \cite{7} further exacerbate the issue by systematically adding redundancy into the test set in order to increase the probability of detection of actual silicon defects. In fact, IBM has found that 70\%–90\% of the logic test patterns can be removed for application specific integrated circuit (ASIC) without any impact to test escapes \cite{8}. According to the international roadmap for device and systems, test costs are now a major part of the manufacturing cost \cite{4}.

A critical first step towards reducing test cost of ICs is to identify the most effective test patterns that contribute to identify faults earlier, and for that reason, test pattern reordering is the key. The objective of testing is to cost-effectively screen chips for minimal test escape (i.e., when faulty circuits are undetected), while at the same time limiting unnecessary yield loss (i.e., inadvertently discarding non-faulty circuits) \cite{8}. It is extremely difficult to do this using traditional test methods under the continuous scaling of ICs technologies. To improve the performance of test, some forms of adaptation are essential.

In recent years, there has been an intensive effort to develop alternative test approaches that can replace traditional test approaches. Therefore, an adaptive testing strategy is now being advocated. In adaptive testing, certain ineffective patterns are dropped so as to shrink test time \cite{9}; the test order, meaning that patterns with higher fail rates are moved forward and applied first so as to save test time \cite{10}. In this way, adaptive testing selects only the most effective test patterns, resulting in significantly reduced test time.

Different approaches have been proposed to reduce test cost. In particular, with the rapid development of machine learning technology, it has been proposed to tackle the job. To reduce test time, Singh et al. \cite{11} used spatial defect clustering information and the available test results for neighboring dies to optimize test costs for VLSI testing. Maxwell et al. \cite{12} dynamically and cost-effectively tailored IC testing to discriminate multiple variations in the manufacturing process. Hapke et al. \cite{13} proposed new fault detection methods using critical area methods. Takashi et al. \cite{14,15} and Xin et al. \cite{16} used machine learning methods in adaptive testing. The above literatures

---

\(^1\) School of Electronic Science and Applied Physics, Hefei University of Technology, Hefei, 230009, China
\(^2\) Department of Electrical Engineering, Anhui Polytechnic University, Wuhu 241000, China
\(^3\) Department of Computer Science and Technology, Pingdingshan Polytechnic college, Pingdingshan,467000, China
\(^a\) huagulg@hfut.edu.cn

DOI: 10.1598/elex.20200420

Received December 4, 2020
Accepted December 15, 2020
Publicized December 23, 2020
Copyedited January 25, 2021
primarily focused on selecting optimal test patterns for reducing test costs, but they did not consider the accuracy of the prediction model, and therefore suffered from poor test quality. To address this problem, Grady et al. [17] presented two applications of “test pattern sampling” for logic testing. Bingjun et al. [18] proposed a novel deep learning method for adaptive test; and Chakrabarty et al. [19, 20] proposed an adaptive testing based on a quality prediction model. Cheng-wen et al. [21] proposed a wafer-level chip-scale package method. However, although these approaches considered the trade-off between test time and test quality, due to the lack of a mathematical model, they were not scalable because the test patterns to be grouped together needed human selection and domain knowledge from chip experts. Therefore, while they offer better precision, they do not address broader applicability and generality. To address this problem, Sabyasachi et al. [22] proposed adaptive testing in which finite state machine models of mixed-signal/RF circuits were abstracted from hardware measurements on fabricated devices. Shambhu et al. [23] adaptively assigned the cores in parallel to the test access mechanisms with variable widths. However, these last two approaches increased additional hardware overhead and manufacturing cost.

As we can see, there has been relatively little research on identifying effective test pattern methods and on selecting the optimal patterns by means of mathematical models [24, 25], which could find the optimal points and achieve broader applicability and generality. Recently, machine learning has achieved immense success in classification related tasks spanning different applications [26, 27, 28]. Motivated by this success, we using SPF algorithm combined with KNN algorithm to address the test pattern reordering task. KNN is used to predict the weight of each test pattern and SPF is to find the most important weight as the shortest path. Our main contributions can be summarized as follows:

1) A novel approach for test patterns reordering using SPF-KNN algorithm is proposed.

2) Optimal test patterns can be selected by means of polynomial regression function, and the optimization problem is converted into a mathematical function to give it broader applicability and generality.

3) The method in this paper is based on software and statistical data. No additional hardware is required, and the complexity and hardware cost are not increased.

2. Preliminaries

In ATE test, where stop-at-first-fail is employed. The idea is to apply more effective patterns earlier to make defective chips quickly fail. As you can see from Fig. 1, the traditional test reordering method is shown on the left, and the test reordering method is shown on the right. The green color “P” stands for testing pass and red color “F” stands for testing fail. In order to achieve the biggest time savings, adaptive test reorders the test item as is shown in the right part in Fig. 1, rank forward the item which are most likely to fail, and the most effective test item can be executed first [29].

3. Improved SPF algorithm

The goal of this algorithm aims to reduce the test cost, and this paper presents an improved SPF algorithm for selecting the most effective test pattern as the shortest path. This improved SPF combines with KNN to reorder the test pattern. Since the weights of all test patterns are randomly and have no specific rules, and machine learning is needed to predict the weight of each test pattern. In this paper, the pattern that fail earlier has the higher weight. Therefore, we observe the change in the weight of the test pattern and construct the prediction model through the weight sequence from the training data, and then predict the weight change of the future test patterns. By means of this improved SPF algorithm, we can reorder the test pattern and apply more effective patterns earlier to make defective chips quickly fail.

SPF algorithm calculates the shortest path from a starting node/vertex to all other nodes as is shown in Fig. 2. The algorithm creates a tree of the shortest distances from the starting node to all nodes in the graph. Finding the shortest path means to find the most effective test patterns.

Step 1 is to create a list of the unvisited nodes in Fig. 3. We haven’t visited any nodes yet, so initially the unvisited list will contain all of the nodes in the graph.

Step 2 is to create a table of the distance from the starting node to each of the node in the graph. The only distance we know at first is the distance from the starting node itself. We’ll then set the distance for all other nodes to infinity, and as we loop through the algorithm the central concept is that we’ll calculate new distances to the nodes along various paths and we replace a node’s distance with a lower value whenever we find one.

Step 3 is the beginning of the main loop of the algorithm. We select as the current node the one that has the shortest distance stored for it.

Then we iterate through each of the neighbors of the cur-
rent node and calculate the distance from the starting node to this neighbor node on the path through our “current node.” In other words, we add the current node’s stored distance to the distance from the current node to the neighbor. We compare the new calculated distance to the distance we already have for each neighbor, and update the distance to the minimum of these two values.

Step 4 is to simply remove the visited node from the unvisited list.

Step 5 is to check whether we have done, and if so break out of the main loop. There are three possible conditions that can terminate the algorithm:
1) We’ve visited all of the nodes.
2) We’ve visited the destination node.
3) The shortest distance for any of the remaining unvisited nodes is infinity. (This will happen if there are nodes that are not connected in any way to the starting node.)

The proposed algorithm is shown in the following steps:

Algorithm SPF: Finding shortest path, that is, to find the most effective test pattern;

1 function Dijkstra(Graph, source):
2 dist[source] ← 0 // Initialization
3 create vertex set Q
4 for each vertex v in Graph:
5 if v ≠ source
6 dist[v] ← INFINITY // Unknown distance from source to v
7 prev[v] ← UNDEFINED // Predecessor of v
8 Q.add_with_priority(v, dist[v])
9 while Q is not empty: // The main loop
10 u ← Q.extract_min() // Remove and return best vertex
11 for each neighbor v of u: // only v that are still in Q
12 alt ← dist[u] + length(u, v)
13 if alt < dist[v]
14 dist[v] ← alt
15 prev[v] ← u
16 Q.decrease_priority(v, alt)
17 return dist, prev

Table I Preliminary number of defective circuits

| Benchmark | Gates | Patterns | Circuits |
|-----------|-------|----------|----------|
| s5378     | 1004  | 258      | 5000     |
| s9234     | 2017  | 379      | 5000     |
| s13207    | 2573  | 480      | 5000     |
| s15850    | 3448  | 437      | 5000     |
| s38417    | 8709  | 919      | 5000     |
| s38584    | 11448 | 653      | 5000     |

4. Experimental results

4.1 Fault simulation environment setup

Because the algorithm proposed in this paper is based on statistics, for a new IC to be tested there must be a certain amount of training data. Therefore, the main task of initialization is to obtain enough data. During the initialization step, circuits must be randomly selected. To better mimic a real-world scenario, circuits are selected randomly from different batches and wafers as initialization chips. The purpose of random selection is to ensure that chips in different batches and different wafers are likely to be tested, so as to achieve a more representative selection. Therefore, in this synthesis, we use independent random variables to simulate the physical states of chips. We also use randomly generated non-linear functions to determine all of the chip measurements from the physical states and add Gaussian noise to mimic a real-world scenario for measurement data.

In the experiment, s5378, s9234, s13207, s15850, s38417, and s35932 of ISCAS 89 benchmark circuits (Digital Circuits) were used as test circuits, as shown in Table I. The test patterns for the circuits were generated by the automatic test pattern generation tool, and C language was used for the circuit logic simulation experiment. First, an initialization step was performed to randomly generate 10 wafers, each of which contained 100 circuits to be tested, and then we randomly selected one gate in each circuit to be tested to be randomly injected with a stuck-at 0 or stuck-at 1 fault as the initialization circuit. A classification model was then established to perform a preliminary sorting of the test set. To verify the performance of the selected test set, 50 wafers were randomly generated, with each wafer containing 100 circuits to be tested, and a gate randomly selected in each circuit to be tested was randomly injected with a stuck-at 0 or 1 fault as an initialization. In order to evaluate the efficiency and accuracy of the SPF-KNN method, the test patterns generated from ATPG are imported into Python as original data. The result of the test patterns after selection is obtained by the machine learning tool Python. Machine learning experiment adopted Spyder 3.7 and scikit-learn library, random seed selection is 0.

4.2 Correlation coefficient as training data

We compare the correlation of the test patterns and get their correlation coefficients as training data. The larger the value of the correlation coefficient, the stronger the correlation. The correlation coefficient is convenient to find the pattern which identify faults earlier and provides a theoretical basis for the SPF-KNN algorithm.

An approach is to visualize the correlation matrix directly,
providing a holistic view over the variable space (Fig. 4). In this representation, each matrix cell denotes the correlation of one variable pair. The matrix view has found a wide set of applications. In our case it uses a matrix-based visualizer to provide an overview of the ranking of features. Many of these methods support matrix reordering. The perception of the clusters can be further enhanced by coding correlation strength to color, yielding a heatmap [24]. However, brightness and color are poor visual variables for quantitative information—spatial variables such as size and proximity are far better choices. Due to the large amount of data, the heatmap of ten test patterns is used to explain.

4.3 Determine the optimal parameter value
In order to make the model achieve higher prediction accuracy, the best ratio of training data and testing data is needed. Therefore, the training data are selected randomly but different from each other, and the test data are collected from the rest of the sampled data. The ratio of the size of training data to that of the test data are 1:9, 2:8, 3:7, 4:6, 5:5, 6:4, 7:3, 8:2, and 9:1. The sampling is repeated 10 times using KNN algorithm. As is shown in Fig. 5. The result might expect that the larger the size of training data, the smaller the value of RMSE for test data. In machine learning, the smaller value of the RMSE, the better for learning.

Fig. 5 shows that the value of RMSE is preferably at 0.42 and ratio at 8:2. It is obvious that the ratio of 8:2 is more robust than 9:1. Therefore, 80% of test patterns as training data and 20% of test patterns as testing data were chosen in this work.

Another parameter that will affect accuracy is the K value. Therefore, the value of K from 1 to 40 is selected to obtain the corresponding accuracy. The sampling is repeated 10 times and the mean accuracy as is shown in Fig. 6. It is easy to find that the highest accuracy is 96.7% when K is 2.

In order to evaluate the efficiency and accuracy of the SPF-KNN method, the test patterns generated from ATPG are imported into Python as original data. The result of the test patterns after selection is obtained by the machine learning tool Python. Machine learning experiment adopted Spyder 3.7 and scikit-learn library, random seed selection is 0.

4.4 Comparison of other test flow
In order to show the superiority of the proposed method, this work compared the proposed method in the last three years. In terms of accuracy, test time, select test patterns and the number of test escape, and the compare results are shown in Table II [30]. Each pattern has the identical length in our experiments, the test time reported is proportional to the number of patterns applied.

From Table II we can see the traditional method has the least number of test escape. However, it spent the most test time. It is easy to find that the proposed method achieved 54.2 seconds time saving but leads to 9 test escapes increasing compared with traditional methods. The reorder-based Ref. [3] and Ref. [2] shown that they have the same test escapes more or less as the proposed method. Nevertheless, there are more test time than proposed method, since they only consider the reduction of test escape but ignore the increase of test time. Moreover, the predictive accuracy of proposed method is higher than other test flow. Therefore, it is the best compromise between test escape and test cost of proposed method.

4.5 Trade-off between test time and test escape
In order to find the optimal point between Test Time and Test Escape. Fig. 7 shows the Test Time and Test Escape metrics based on the proportion of test patterns. The Test
Fig. 7  Optimal value between test escape and test time.

Test escape decreases from 279 to 34, while Test Time increases from 51 to 190 as the test pattern increases from 10% to 90%. When the proportion of test patterns equals 60%, the trade-off between Test Escape and Test Time can be selected from the results, and the optimum strategy can be selected for the purpose of tuning Test Escape and Test Time.

To better demonstrate the effectiveness of adaptive test, and find the optimal ratio of patterns using mathematical functions, this work converts the problem of finding the optimal ratio into a mathematical function to give it broader applicability and generality. Fig. 8 shows the test escape polynomial regression curve based on eight different degrees. As can be seen from the eight pictures, when degree = 5 (fifth-degree polynomial), the effect of image fitting is significantly better than other results. However, as the degree increases, the curve shows obvious oscillation, which is the phenomenon of overfitting. The parameter results and function obtained when degree = 5 is

\[
y(x) = 640.984x^5 + 562.456x^4 - 2838.089x^3 + 2347.678x^2 - 1006.364x + 358.333
\]

Fig. 8  Test escape polynomial regression based on different degrees.

4.6 Searching optimal patterns for multiple sets of data
This work further pursues the characteristics for large amounts of data, and as a result, more effective points were selected; 10%–90% of the test patterns were used; and the random resampling was repeated 100 times. Based on the least square method, the fitted surface of 100 sets of data between test escape and test time is shown in Fig. 9, and the corresponding function is

\[
z = -0.002xy + 0.290x + 0.301y - 9.832
\]

where \(z\) is the corresponding ratio of patterns, and \(x\) and \(y\) represent the test time and the number of test escapes, respectively. The optimal number of test patterns can be calculated from this function.

Taking the above functions into consideration, this work determines the optimal test patterns according to different defect levels and cost requirements, which shows that this mathematical function method can update the mathematical function according to the statistical data after each test of data, and update the test patterns simultaneously, which shows the method’s adaptive ability.

4.6 Searching optimal patterns for multiple sets of data
This work further pursues the characteristics for large amounts of data, and as a result, more effective points were selected; 10%–90% of the test patterns were used; and the random resampling was repeated 100 times. Based on the least square method, the fitted surface of 100 sets of data between test escape and test time is shown in Fig. 9, and the corresponding function is

\[
z = -0.002xy + 0.290x + 0.301y - 9.832
\]

where \(z\) is the corresponding ratio of patterns, and \(x\) and \(y\) represent the test time and the number of test escapes, respectively. The optimal number of test patterns can be calculated from this function.

Taking the above functions into consideration, this work determines the optimal test patterns according to different defect levels and cost requirements, which shows that this mathematical function method can update the mathematical function according to the statistical data after each test of data, and update the test patterns simultaneously, which shows the method’s adaptive ability.

5. Conclusion
Our work is an innovative test cost reduction method, which selects the effective test patterns with improved SPF-KNN algorithm. It will make higher fail rate pattern moved forward and applied first to save test time. The comparison results show that the proposed idea significantly improves the test time and test efficiency compared to conventional test flows, which can provide test cost reduction without increasing the defect level obviously. Furthermore, this algorithm is completely software-based and does not require any additional hardware overhead. However, the bad sampling may lead to inaccurate predictions or increased test escape. With the growing complexity of IC, more data will be used, and the sampling will be more accurate. In addition, the reordering method can update the prediction model according to the statistical data and update the test patterns simultaneously, which shows the method’s adaptive ability.

Acknowledgment
This work was supported in part by the Natural Science Foundation of China under the Granted No. 62027815, 61834006, 61874156 and 61904001.

References

[1] M.B. Alawieh, et al.: “Wafer map defect patterns classification using deep selective learning,” 2020 57th ACM/ESDA/IEEE De-
sign Automation Conference (DAC) (2020) (DOI: 10.1109/dac18072.2020.9218580).

[2] T. Song, et al.: “Novel application of deep learning for adaptive testing based on long short-term memory,” 2019 IEEE 37th VTS (2019) (DOI: 10.1109/vts.2019.8758628).

[3] H.-G. Stratigopoulos and C. Streitwieser: “Adaptive test with test escape estimation for mixed-signal ICs,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 37 (2018) 2125 (DOI: 10.1109/TCAD.2017.2783302).

[4] Semiconductor Industry Association: “International roadmap for devices and system (IRDS),” (2020).

[5] T. Song, et al.: “Pattern reorder for test cost reduction through improved SVRMARk algorithm,” IEEE Access 8 (2020) 147965 (DOI: 10.1109/access.2020.3016039).

[6] S.C. Ma, et al.: “An experimental chip to evaluate test techniques experiment results,” Proc. Intl. Test Conf. (1995) 663 (DOI: 10.1109/test.1995.529895).

[7] K.Y. Cho, et al.: “Gate exhaustive testing,” Proc. Intl. Test Conf. (2005) 1 (DOI: 10.1109/test.2005.1584040).

[8] F.-F. Ferhani, et al.: “How many test patterns are useless?,” Proc. VLSI Test Symp. (2008) 23 (DOI: 10.1109/vts.2008.27).

[9] S. Gupta, “Adaptive online testing for efficient hard fault detection,” 2009 IEEE International Conference on Computer Design (2009) 343 (DOI: 10.1109/iccd.2009.5413132).

[10] P. Maxwell, “Adaptive test directions,” 2010 15th IEEE European Test Symposium (2010) 12 (DOI: 10.1109/etsym.2010.5512789).

[11] A.D. Singh and C.M. Krishna: “On optimizing VLSI testing for product quality using die-yield prediction,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 12 (1993) 695 (DOI: 10.1109/43.277614).

[12] P. Maxwell, “Adaptive testing: dealing with process variability,” IEEE Design Test Comput. 28 (2011) 41 (DOI: 10.1109/mdt.2011.118).

[13] F. Hapke and P. Maxwell: “Total critical area based testing,” 2018 IEEE ITC (2018) 1 (DOI: 10.1109/TEST.2018.8624825).

[14] R. Pan, et al.: “Black-box test-coverage analysis and test-cost reduction based on a Bayesian network model,” 2019 IEEE 37th VLSI Test Symposium (VTS) (2019) 1 (DOI: 10.1109/vts.2019.8758639).

[15] R. Pan, et al.: “Black-box test-cost reduction based on Bayesian network models,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. (DOI: 10.1109/TCAD.2020.2994257).

[16] M. Shintani, et al.: “A variability-aware adaptive test flow for test quality improvement,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 33 (2014) 1056 (DOI: 10.1109/TCAD.2014.2305835).

[17] M. Grady, et al.: “Adaptive testing - cost reduction through test pattern sampling,” 2013 IEEE International Test Conference (ITC) (2013) 1 (DOI: 10.1109/test.2013.6651891).

[18] B. Xiao, et al.: “Novel applications of deep learning hidden features for adaptive testing,” 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC) (2016) 743 (DOI: 10.1109/aspcad.2016.7428100).

[19] M. Liu, et al.: “Fine-grained adaptive testing based on quality prediction,” 2018 IEEE ITC (2018) (DOI: 10.1109/TEST.2018.8624891).

[20] M. Agrawal and K. Chakrabarty: “Test-cost modeling and optimal test-flow selection of 3-D-stacked ICs,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 34 (2015) 1523 (DOI: 10.1109/tdcad.2015.2492277).

[21] K.-L. Wang, et al.: “Test cost reduction methodology for InFO wafer-level chip-scale package,” IEEE Design Test 34 (2017) 50 (DOI: 10.1109/mdat.2016.2562060).

[22] S. Deyati, et al.: “Adaptive testing of analog/RF circuits using hardware extracted FSM models,” 2016 IEEE 34th VLSI Test Symposium (VTS) (2016) 1 (DOI: 10.1109/vts.2016.7477283).

[23] D. Zhao and S. Upadhyaya: “Dynamically partitioned test scheduling with adaptive TAM configuration for power-constrained SoC testing,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 24 (2005) 956 (DOI: 10.1109/tdcad.2005.847893).

[24] W. Zhan and Z. Shao: “Test patterns reordering method based on gamma distribution,” Integration 72 (2020) 66 (DOI: 10.1016/j.inTEGRATION.2019.12.008).

[25] R. Tibshirani: “Regression shrinkage and selection via the lasso,” Journal of the Royal Statistical Society B 58 (1996) 267 (DOI: 10.1111/j.2517-6161.1996.tb02080.x).

[26] A.E. Hoerl and R.W. Kennard: “Ridge regression: biased estimation for nonorthogonal problems,” Technometrics 12 (1970) 55 (DOI: 10.1080/00401706.1970.10488634).

[27] H. Zou and T. Hastie: “Regularization and variable selection via the elastic net,” Journal of the Royal Statistical Society B 67 (2005) 301 (DOI: 10.1111/j.1467-9868.2005.00503.x).

[28] T. Ni, et al.: “Architecture of cobweb-based redundant TSV for clustered faults,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 28 (2020) 1736 (DOI: 10.1109/TVLSI.2020.2995094).

[29] T. Song, et al.: “Pattern reorder for test cost reduction through improved SVMRANK algorithm,” IEEE Access 8 (2020) 147965 (DOI: 10.1109/access.2020.3016039).

[30] C. Xue and R.D. Blanton: “Test-set reordering for improving diagnosability,” 2017 IEEE 35th VLSI Test Symposium (VTS) (2017) 1 (DOI: 10.1109/VATS.2017.7928926).