Quest for a Quantum Search Algorithm for Testing Stuck-at Faults in Digital Circuits

Muralidharan Venkatasubramanian
Ph.D Candidate
Department of Electrical and Computer Engineering,
Auburn University,
Auburn, Alabama 36849
Email: vmn0001@auburn.edu

Vishwani D. Agrawal
James J. Danaher Professor
Department of Electrical and Computer Engineering,
Auburn University,
Auburn, Alabama 36849
Email: vagrawal@eng.auburn.edu

***Abstract*—It is colloquially known that searching for test vectors to test the last few hard to detect stuck-at faults is computationally most expensive and mathematically NP-complete. Due to the complex nature of this problem, attempts made to successfully test a digital circuit for all faults in computational linear time start becoming exponential with an increase in circuit size and complexity. Various algorithms have been proposed where new vectors are generated by using previous successful vectors with similar properties. However, this leads to a bottleneck when trying to find hard to detect stuck-at faults which have only one or two unique tests and their properties may not match other previously successful tests. We propose a new unique algorithm that attempts to vastly improve the test search time for these few hard to detect faults by classifying all test vectors in the vector space in three categories: Category I vectors that activate the desired stuck-at fault but may not propagate it to the primary outputs (POs), Category II vectors that propagate the fault site value to the POs, and Category III vectors that neither activate nor propagate the fault. By bounding our search to vectors in categories I and II, and avoiding category III vectors, it is easier to arrive at the solution faster than other algorithmic implementations. The final solution itself lies in the intersection of categories I and II vectors, and it is easier to search for a test vector in a smaller subset of the large vector space. We have demonstrated the proof of concept and detailed working of our algorithm by comparing it with a random test generator.**

I. Introduction

A circuit with n primary inputs (PIs) can have 2^n possible test vectors. If a fault is present in that circuit, a test vector to correctly detect that fault has to be present within those 2^n possible combinations. Hence, the problem of VLSI Testing can be rephrased as a database search problem. Classic algorithms like the D-Algorithm [18], PODEM [11], and FAN [9] have long been established as foundations over which other algorithms [6], [13], [20] have built upon to vastly improve the search time for test vectors of stuck-at faults. While the D-Algorithm is a depth-first search algorithm, PODEM and FAN have their roots in breadth-first search.

As circuits became more complex and test generation time started increasing due to the NP complexity of VLSI Testing [10], [14], [21], other algorithms were proposed which derive and extract new test vectors based on the properties of previous successful vectors. Some of the more prominent examples are weighted-random test generators [1], [19], test generation using spectral information [22], and anti-random test pattern generation [15] amongst others which are all different variations of genetic search algorithms.

However, since these algorithms worked by extracting new vectors based properties similar to previous successes, all these algorithms start hitting bottlenecks when trying to test hard to detect faults that may have just a handful of unique tests in the entire search space. It was because these hard to detect faults had vectors which had different properties as compared to these previous successful vectors and hence the vector search time devolved back to the classic NP-hard problem of VLSI Testing.

Current testing algorithms have shown tremendous resilience in finding test vectors and aiming to achieve 100% fault coverage. However, a growing interest in quantum computing has spurred investigations in the areas of probabilistic computing algorithms [3], [5] leading to certain problems (especially NP complete problems) being revisited to try to find an optimal solution in linear time.

This paper proposes a unique algorithm that aims to clear the final bottleneck by attempting to implement Grover's Algorithm for database search [12] in the area of VLSI testing. While classical search algorithms like depth-first, breadth-first, genetic algorithms, etc., [7] which have been extensively used to design testing algorithms (as highlighted above) can complete a query in only linear time, Grover's algorithm can search through the database in sub-linear time ($O\sqrt{N}$) and can be deemed the fastest way possible to search theoretically [4].

The key component behind Grover's Algorithm is the idea of an "Oracle" which can recognize the solution to a database search problem without knowing the actual solution. The concept of the oracle rests on the important fact there is a distinction between *"knowing"* the solution and *"recognizing"* the solution and it is possible to do the latter without knowing the former [17].

Contemporary testing algorithms try to arrive at the solution in a deterministic manner by trying to propagate the

978-1-4799-8606-4/15/$31.00 ©2015 IEEE
IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS), Amherst, MA, Oct. 12-14, 2015

fault to the primary outputs (POs) or by generating new test vectors based on previous successes based on the corollary that there is a strong correlation between the parallel bits of test vectors applied at PIs [2]. However, by ignoring the failed test vectors, these algorithms are throwing away lots of potentially useful information which can help deduce the solution faster. We attempt to answer this question: **"How to design a new test algorithm which utilizes the information from failed attempts effectively?"**

Our conjecture is that by moving away from the test vectors similar in properties to these failed test vectors, we will arrive at the solution in quicker iterations than current algorithms. Also, since we are now utilizing both successful and failed test vector information, we can hypothesize that our algorithm contains the essence of Grover's algorithm. This major key point differentiates our algorithm with all the current algorithms in the market, as nobody has yet managed to successfully utilize the information of failed test vectors to try and generate new test vectors.

This paper is divided into five more sections. Section II expands on the working and explanation of our proposed algorithm, Section III highlights how our simulations were conducted and what sort of software and tools were used, Section IV discusses and elaborates the results obtained from our simulations along with graphs and figures. Sections V and VI highlight our future direction of research which can be undertaken and the conclusion.

II. Algorithm Outline and Implementation

A. Conceptual Outline

The proposed algorithm primary conjecture is that by avoiding all vectors with properties similar to known failed test vectors, the direction of the search gets skewed towards the correct test vector. To aid the algorithm in the search, the test vectors in the vector space have been classified in three major categories:

- **Activation vectors:** These vectors activate a desired stuck-at fault on the fault line of a circuit. However, not all vectors may propagate the fault to POs. For example, if a line in a circuit is stuck-at-1, these vectors will activate that fault. However, it is possible that the fault may never get propagated to the PO because no path is sensitized.
- **Propagation vectors:** These vectors will sensitize the path to POs and propagate a desired line's fault to the POs. In other words, if any stuck-at fault is placed on a particular line, the vectors in this category will propagate both fault types to the POs.
- **Failed vectors:** These vectors neither activate the fault site to the desired stuck-at fault nor sensitize the path to propagate the vectors to the POs. These vectors only provide information on what to avoid and bound our search in the subsets of "activation and propagation regions" of the vector space.

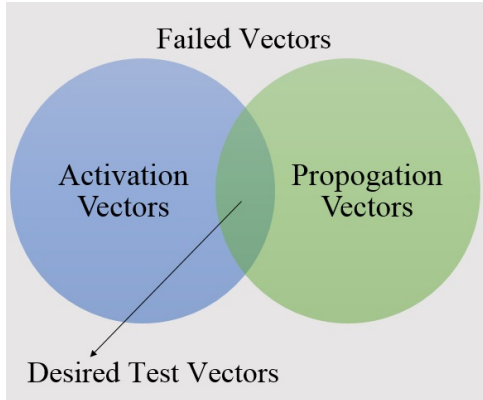


Fig. 1. All test vectors in the vector space classified in appropriate categories for a given stuck-at fault.

The ideal test vector will not only activate the desired stuck-at fault but sensitize a path to propagate it to the POs as well. As Fig. 1 further highlights, the correct test vector lies at the intersection of the activation vector region and propagation vector region. However, hard to detect faults may have only one or two such unique vectors. It is easier to find vectors which can either activate the fault but do not sensitize a path or conversely sensitize a path but not activate the fault. These vectors have partially useful information which can be used to hone into the correct solution steadily. The failed vectors restrict our search in the region of "partial desirability" and hence act as a fence so that we do not search outside of those constraints.

B. Implementation

The algorithm's concept outlined in the previous section can be implemented in a variety of ways or techniques. This paper utilizes the the method of skewing the independent weighted probabilities of the PIs in such a manner so that the search moves away from the failed test vector region. To put it simply, we postulate if a certain probability weight at the PI generates a logic value (1 or 0) which neither activates the fault nor sensitizes a path to the POs, it is best to invert the probability weight before generating a new value for that line. This method is repeated till the search enters the region of activation vectors and/or propagation vectors (colloquially, region of "partial desirability").

Once the search enters either of these regions, it is in our best interest that the search does not deviate away from this subset of vector space. From here on now, the weighted probability of the failed vectors is used to modify the weighted probability of the vectors in the partial desirability region. These modifications are made in smaller increments in order to make smaller steps towards the correct test vector. Figure 2 gives a detailed flowchart of the implementation process of the concept highlighted in the previous section.

Figure 3 illustrates a generic example of how our algorithm skews the probability of the desired test vectors as compared

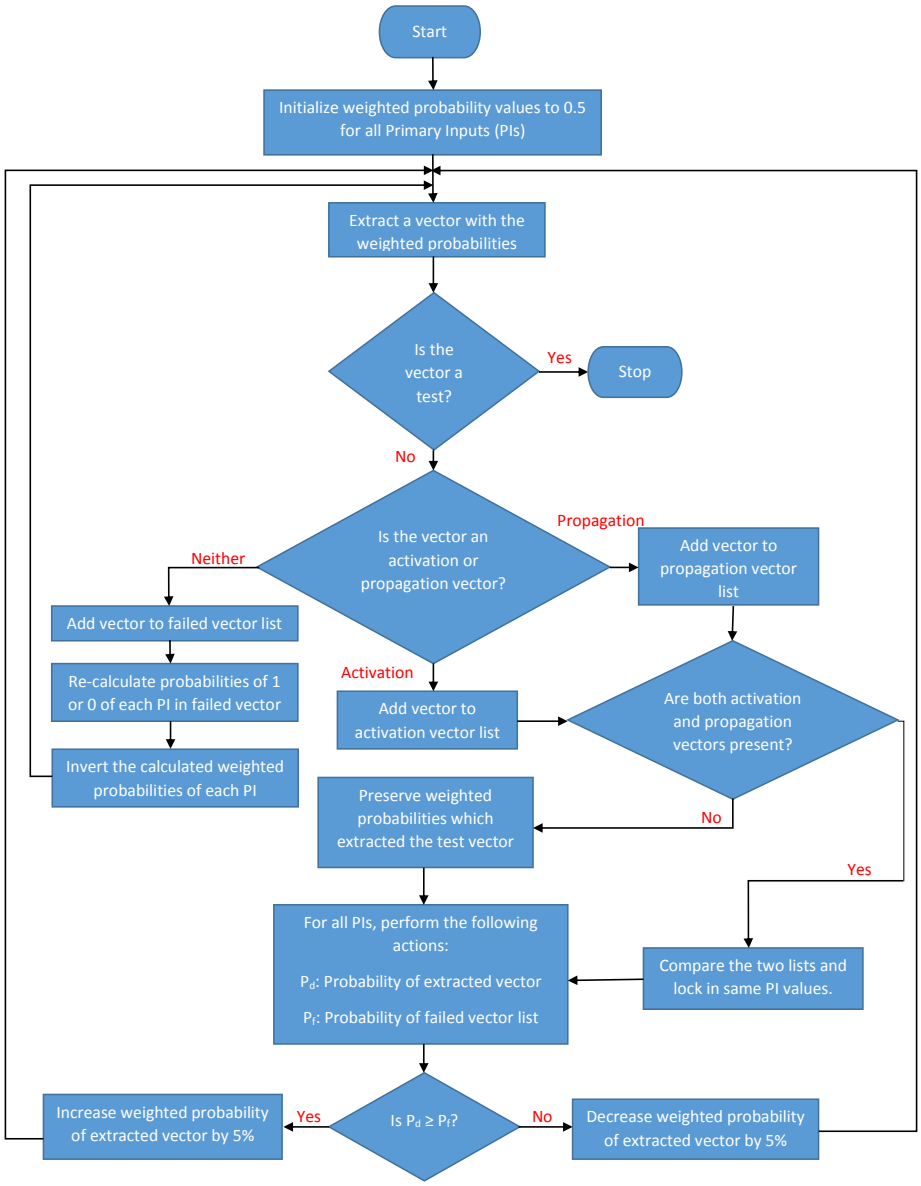


Fig. 2. Flowchart of the implementation of skewing the search in the vector space using weighted probabilities.

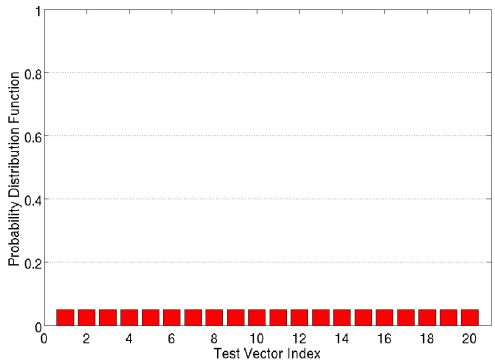
to a random search without replacement algorithm. Figure 3(a) highlights the initial case when no tests have been found. This illustrates that all test vectors have equal probability amplitude in the beginning. In a random search algorithm, if a vector is not a test, it's probability amplitude becomes 0 for the next iteration and its previous amplitude is equally distributed among the remaining test vectors. Figures 3(c) and 3(e) illustrate this phenomenon after 10 iterations and 15 iterations respectively. Because, the probabilities of all the vectors go up equally, it can take a lot of trials for the solution's probability to be appreciable and picked.

Figure 3(b) highlights the initial case when no tests have been found for our proposed algorithm. As expected, it is the same as the case of random search initially. However, each failure helps skew the search in the direction of the desired test vector. Since the probabilities of each vector changes according to previous failures, the probability amplitude of the desired test vector increases faster as compared to other vectors. Figures 3(d) and 3(f) illustrate this phenomenon after 10 iterations and 15 iterations, respectively.

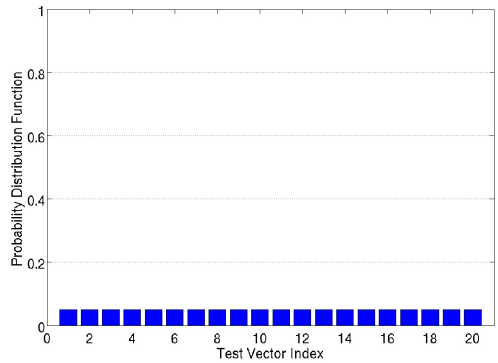
III. Simulation Setup

The RTL model written in Verilog was used to perform the simulations. Figure 4 illustrates the gate level design of the circuit which was used to test the proposed algorithm. The red 'X' highlights the fault site where the stuck-at-1 fault is present for which a test vector needs to be generated.

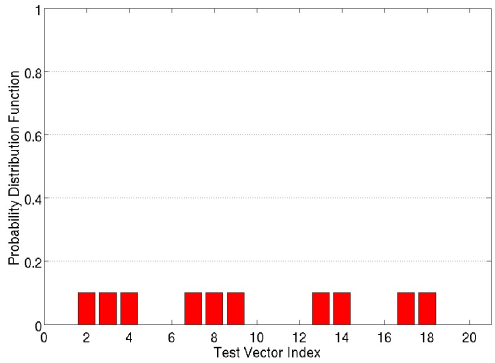
The fault simulator used was FastScan [8] from Mentor Graphics. The random pattern generator built into FastScan was used to emulate a random search of test vectors for desired



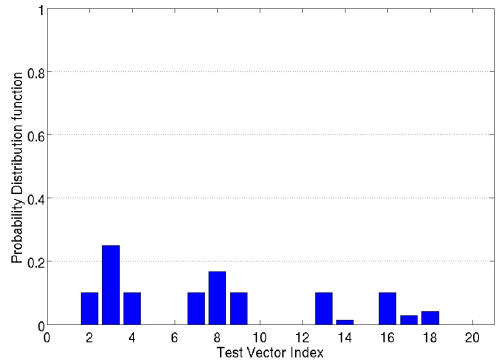
(a) Initial P.D.F. of test vectors of random search algorithm



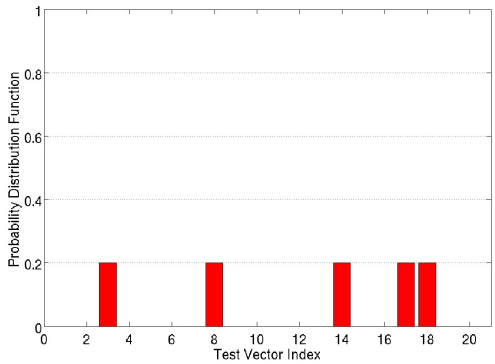
(b) Initial P.D.F. of test vectors of proposed algorithm



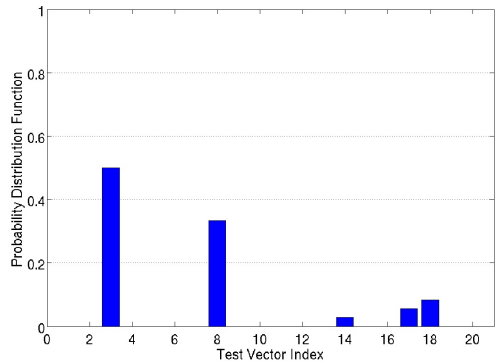
(c) P.D.F. of test vectors after 10 random search iterations



(d) P.D.F. of test vectors after 10 proposed algorithm iterations



(e) P.D.F. of test vectors after 15 random search iterations



(f) P.D.F. of test vectors after 15 proposed algorithm iterations

Fig. 3. Probability density function (P.D.F.) values of random search and proposed algorithm with increase in iterations. (a), (c), and (e) depict the P.D.F. changes of random search while (b), (d), and (f) depict the P.D.F. changes of our proposed algorithm.

stuck-at fault in the circuit.

The proposed algorithm was written and coded in MATLAB [16] provided by MathWorks. The test vectors extracted from MATLAB was sent to FastScan for verification i.e. to check if the vector can test the fault in the circuit. This was done using the fault simulator mode in FastScan.

IV. Results and Discussion

Figure 4 shows a circuit with a stuck-at-1 fault on a line. Since it has 6 PIs, there can be a maximum of 64 test vectors in the vector space. Out of these, only one ("111111" on the PIs) can successfully test the stuck-at-1 fault on that line. The fault has eight activation vectors ("111000, 111001, 111010, 111011, 111100, 111101, 111110, 111111") and four propagation vectors ("001111, 011111, 101111, 111111").

A random search algorithm will randomly pick a test vector from the vector space till the correct test vector is found. Upon running 100 trials of a random search algorithm on the example circuit, it was seen that random search needs an average of 34 iterations to find the test for the given stuck-at fault. Random search had a best case search of 1 iteration and worst case search of 64 iterations. Our proposed algorithm needed an average of 14 iterations with a best case of 1 iteration and worst case of 38 iterations. Table I illustrates these results.

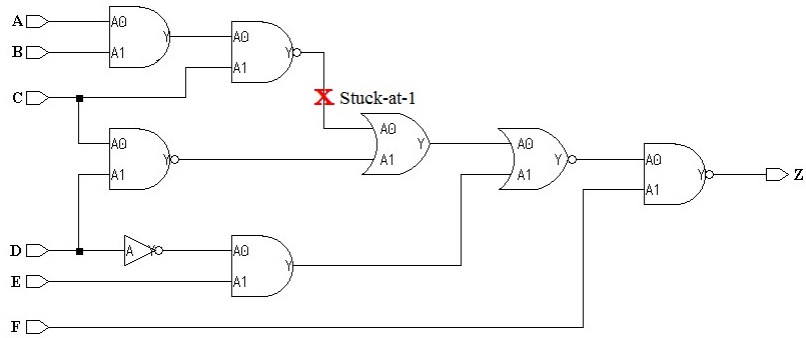


Fig. 4. Gate level design of the circuit used to test the algorithm. 'X' marks the stuck-at-1 fault for which the test vector was to be found.

TABLE I. COMPARISON OF AVERAGE, BEST CASE, AND WORST CASE ITERATIONS NEEDED TO SEARCH FOR DESIRED TEST VECTOR BETWEEN RANDOM SEARCH WITHOUT REPLACEMENT AND PROPOSED ALGORITHM.

Algorithm type	Average iterations	Best case iterations	Worst case iterations
Random search	34	1	64
Proposed algorithm	14	1	38

TABLE II. COMPARISON OF IDEAL AND EXPERIMENTAL ITERATION RESULTS OF C6288 (n = 32 PIS) BENCHMARK CIRCUIT NEEDED TO SEARCH FOR DESIRED TEST VECTOR BETWEEN RANDOM SEARCH WITHOUT REPLACEMENT AND PROPOSED ALGORITHM.

Algorithm type	Average iterations for $N = 2^{32}$	
	Ideal	Experiment
Random search	$N/2 = 2.14 \times 10^9$	2.42×10^9
Proposed algorithm	$\sqrt{N}$ = 65,536	74,352

Figure 5 illustrates the distribution of iterations needed to find the test vector over a sample of 100 trial simulations. While both the algorithms have a normal distribution fit, a random search algorithm has iteration values over the entire vector space. Our proposed algorithm performs much better than a random search algorithm with an average of 14 iterations. This is more than a 50% improvement in test search time. In fact, about 67 trials out 100 consistently finish under 16 iterations which is half of the ideal average (32 iterations out of 64) needed to search through the vector space. It is seen that once the search enters the "region of partial desirability" (highlighted by activation vectors or propagation vectors), the search for the correct test accelerates very quickly and the algorithm is able to hone into the right test within a few iterations.

Table II shows a preliminary result of the proposed algorithm compared to a random search for the c6288 benchmark circuit. The circuit has $n = 32$ primary inputs (PIs). Thus, the test search space contains $N = 2^{32}$ = 4,294,967,296 vectors. Hence, Grover's Algorithm should ideally take $\sqrt{N} = 65,536$ iterations on average to find a test. We find the test in 74,352 iterations on average. In contrast, totally random searches took 2.42×10^9 iterations on average, which is comparable to the theoretical value $N/2$. These are just preliminary results which tells us that there are still some ways to go to get the optimal improvement. However, it does validate the proof of the algorithm with bigger circuits and demonstrates how it may scale up.

V. FUTURE WORK

Future directions of this algorithm includes testing the proposed algorithm on all the ISCAS'85 benchmark circuits, i.e., all the combinational benchmark circuits. Another direction of this work lies in comparing this algorithm's efficiency in generating tests to other contemporary algorithms mainly weighted random test generation, anti-random TPG, and spectral test generators. Since the algorithm's core concept of utilizing failed test vectors' information to search for new vectors is abstract, different methods of implementation can be used to further improve search speed. One such technique is the use of vector correlation between the PIs. By utilizing the bit correlation between the primary input lines, it is possible to move away from the undesirable correlation of the failed test vectors and quickly hone into the correct test vector.

VI. CONCLUSION

This paper successfully shows the working of our new algorithm. The proof of concept of our proposed algorithm has been successfully demonstrated, and initial results show that our algorithm performs exceptionally well when compared to random test pattern generators. The future direction of our work is to compare our algorithm with other well-known published works for benchmark circuits and prove its superiority. With Moore's law coming to a crawl, and silicon technology slowly approaching its nadir, research in quantum computing and quantum algorithms is on the ascent. Previously described algorithms like Shor's algorithm and Grover's algorithm are already being used as a test for the working of quantum computers. All probability based algorithms are not quantum algorithms, but all quantum algorithms are probabilistic algorithms. Our end objective is to create a quantum testing algorithm i.e. create a probabilistic algorithm which can run on a quantum computer. To that

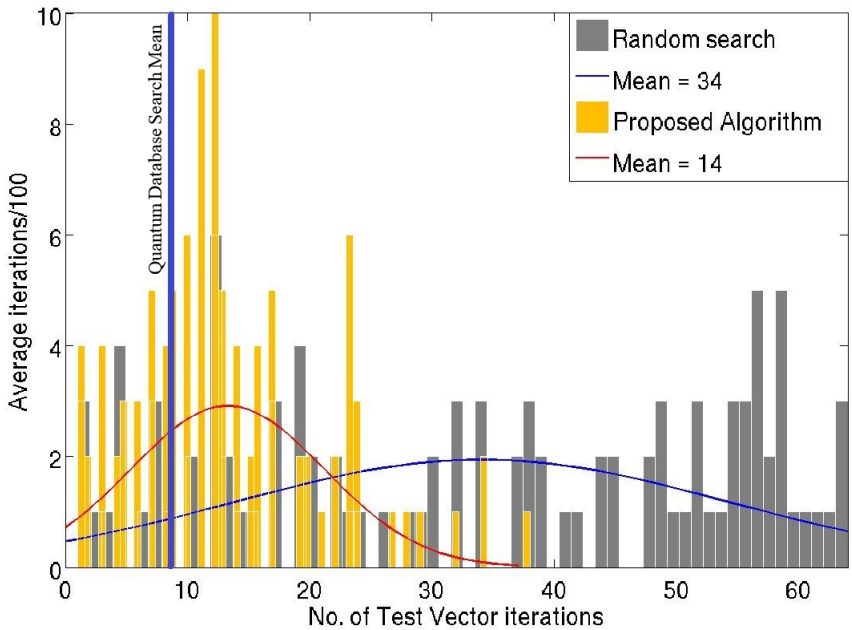


Fig. 5. Distribution of no. of iterations needed to search for the correct test vector. Grey bars indicate the iterations needed for random search without replacement algorithm. The blue line indicates the normal distribution fit with mean = 34. The light orange bars indicate the iterations needed for our proposed algorithm. The red line highlights the normal distribution with mean = 14. The mean value for quantum search of 64-element database is $\sqrt{64} = 8$ iterations.

aspect, we have tried to emulate the mathematical model of Grover's algorithm in a more practical manner for VLSI testing. We might have not yet achieved the goal, but that is the ultimate direction with which we are undertaking a foray in this research.

References

[1] V. D. Agrawal, "An Information Theoretic Approach to Digital Fault Testing," *IEEE Trans. Computers*, vol. 30, no. 8, pp. 582–587, 1981.

[2] S. B. Akers, "Universal Test Sets for Logic Networks," in *IEEE Conference Record of 13th Annual Symposium on Switching and Automata Theory*, 1972, pp. 177–184.

[3] D. Angluin and L. G. Valiant, "Fast Probabilistic Algorithms for Hamiltonian Circuits and Matchings," *Journal of Computer and System Sciences*, vol. 18, no. 2, pp. 155–193, 1979.

[4] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, "Strengths and Weaknesses of Quantum Computing," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1510–1523, 1997.

[5] D. M. Blei, "Probabilistic Topic Models," *Communications of the ACM*, vol. 55, no. 4, pp. 77–84, 2012.

[6] S. T. Chakradhar, V. D. Agrawal, and S. G. Rothweiler, "A Transitive Closure Algorithm for Test Generation," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 7, pp. 1015–1028, 1993.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[8] "TESSENT FastScan." Mentor Graphics. http://www.mentor.com/products/silicon-yield/products/fastscan/.

[9] H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms," *IEEE Trans. Computers*, vol. 32, no. 12, pp. 1137–1144, 1983.

[10] H. Fujiwara and S. Toida, "The Complexity of Fault Detection Problems for Combinational Logic Circuits," *IEEE Trans. Computers*, vol. 31, no. 6, pp. 555–560, 1982.

[11] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," *IEEE Trans. Computers*, vol. 30, no. 3, pp. 215–222, 1981.

[12] L. K. Grover, "A Fast Quantum Mechanical Algorithm for Database Search," in *Proc. 28th Annual ACM Symp. Theory of Computing*, 1996, pp. 212–219.

[13] M. Henftling, H. Wittmann, and K. Antreich, "A Formal Hon-Heuristic ATPG Approach," in *Proc. Conference on European Design Automation*, IEEE Computer Society Press, 1995, pp. 248–253.

[14] O. H. Ibarra and S. Sahni, "Polynomially Complete Fault Detection Problems," *IEEE Trans. Computers*, vol. 24, no. 3, pp. 242–249, 1975.

[15] Y. K. Malaiya, "Antirandom Testing: Getting the Most Out of Black-Box Testing," in *Proc. Sixth IEEE International Symp. Software Reliability Engineering*, 1995, pp. 86–95.

[16] "MATLAB." MathWorks. http://www.mathworks.com/products/matlab/.

[17] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. New York: Cambridge University Press, 10th edition, 2011.

[18] J. P. Roth, "Diagnosis of Automata Failures: A Calculus and a Method," *IBM Journal of Research and Development*, vol. 10, no. 4, pp. 278–291, 1966.

[19] H. D. Schnurmann, E. Lindbloom, and R. G. Carpenter, "The Weighted Random Test-Pattern Generator," *IEEE Trans. Computers*, vol. 24, no. 7, pp. 695–700, 1975.

[20] M. H. Schulz, E. Trischler, and T. M. Sarfert, "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, no. 1, pp. 126–137, 1988.

[21] G. Seroussi and N. H. Bshouty, "Vector Sets for Exhaustive Testing of Logic Circuits," *IEEE Trans. Information Theory*, vol. 34, no. 3, pp. 513–522, 1988.

[22] N. Yogi and V. D. Agrawal, "Spectral RTL Test Generation for Gate-Level Stuck-at Faults," in *Proc. 15th Asian Test Symposium*, IEEE, 2006, pp. 83–88.