iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** V **Month of publication:** May 2021

DOI: https://doi.org/10.22214/ijraset.2021.34837

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue V May 2021- Available at www.ijraset.com

Automatic Test Pattern Generation using Grover's Algorithm

Nishant Agrawal[1], Kumar Shashank[2], Shashank R. S.[3], Rahul J[4], Dr. N. Ramavenkateswaran[5]
[1, 2, 3, 4]*Student,* [5]*Assistant Professor, Department of Electronics and Communication Engineering, R V College of Engineering, Bengaluru, India*

Abstract: ***Quantum computing is an exciting new field in the intersection of computer science, physics and mathematics. It refines the central concepts from Quantum mechanics into its least difficult structures, peeling away the complications from the physical world. Any combinational circuit that has only one stuck at fault can be tested by applying a set of inputs that drive the circuit to verify the output response. The outputs of that circuit will be different from the one desired if the faults exist. This project describes a method of generating test patterns using the Boolean satisfaction method. First, the Boolean formula is constructed to express the Boolean difference between a fault-free circuit and a faulty circuit. Second, the Boolean satisfaction algorithm is applied to the formula in the previous step. The Grover algorithm is used to solve the Boolean satisfaction problem. The Boolean Satisfiability problem for Automatic Test Pattern Generation(ATPG) is implemented on IBM Quantum Experience. The Python program initially generates the boolean expression from the file and converts it into Conjunctive Normal Form(CNF) which is passed on to Grover Oracle and runs on IBM simulator and produces excellent results on combinational circuits for test pattern generation with a quadratic speedup. Grover's Algorithm on this problem has a run time of O(√N).***

Keywords: Automatic Test Pattern Generation (ATPG), Satisfiability, Grover's Algorithm, Qubit, Amplitude Amplification, Oracle, Conjunctive Normal Form

I. INTRODUCTION

A. Introduction to Quantum Computing

The quantum computer was originally conceived as a method to simulate the system of quantum mechanics, which cannot be effectively simulated on a classical computer . The basic idea of quantum computing is that, unlike traditional bits based on transistors, a single qubit can have values of zero and one at the same time called as superposition. There is a probability of measuring zero and a probability of measuring one, but this measurement will collapse the quantum system, producing only zero or one. Since one qubit can represent two values at the same time (zero and one), N qubits can represent 2N values at the same time. In addition to superposition, quantum circuits also use entanglement, which is the phenomenon in quantum mechanics, in which two particles can be entangled. Quantum information processing (QIP) utilizes qubits, two-state quantum-mechanical frameworks that can be in superposition of the two states simultaneously in order to have computational benefits.[1] The physical realization of quantum computers follows the concept of quantum physics and is based on quantum dots, nuclear magnetic resonance, superconducting junction and ion traps. A quantum circuit is a computing mechanism that combines real-time classical processing with coherent quantum operations on quantum data such as qubits. Quantum gates are unitary gates and unlike classical gates, unitary gates are always reversible.

B. Quantum Algorithms

Richard Feynman was the first to suggest the notion of a quantum computer in 1982 with a simple reason that none of the classical systems has abilities to simulate Quantum Mechanical systems[2]. As compared with classical computers, a Quantum computer based on the laws of Quantum Mechanics could simulate these systems more precisely and productively.

David Deutsch and Richard Jozsa proposed the first deterministic quantum algorithm called Deutsch Jozsa algorithm in 1992 with improvements by other researchers in 1998[3]. It was the first quantum algorithm which performed better than deterministic classical algorithms. This algorithm solves f(x) in a single iteration and checks whether the given function is constant or balanced. A function will be considered constant if it returns all 0's or all 1's for any input.

Grover's Algorithm which is also referred to as quantum search algorithm is used for unstructured search to produce a particular output value with high probability. This algorithm uses an amplitude amplification trick to search for a particular element in an unsorted database and hence provides quadratic speedup when compared to its classical counterparts. Grover's Algorithm is presently one of the efficient algorithms for 3SAT i.e. satisfiability problem.[4]Using Grover's Amplification trick, the marked item can be found in around$\sqrt{N}$ steps. Hence providing a quadratic speedup The Grover algorithm is used to solve the Boolean satisfaction problem.

C. Automatic Test Pattern Generation(ATPG)

ATPG is an electrical plan robotization strategy/innovation for deciding an info succession that permits programmed test gear to recognize right circuit conduct and faulty circuit conduct brought about by issues. A test pattern for a potentially defective circuit is a set of inputs that cause the outputs of the circuit to differ depending on whether the circuit is defective or not[5]. We'll need a model of the circuit's likely problems to get the input set (faults). We'll employ the most popular of the existing testing methods, the single stuck-at technique. A defective circuit is anticipated to act as if it were defect-free under this concept, with the exception of one wire connected to either a logic 0 or a logic 1.

Automatic Test Pattern Generation (ATPG) is an important topic in VLSI testing since it enables a designer to automatically create tests to check for manufacturing problems in a design. As a result, the tool is approximately 30% faster than a commercial ATPG solution.Automatic test pattern generation systems differentiate faulty from defect-free components by producing input sets that cause the outputs of a component under test to differ depending on whether the component is defective or not. The Boolean Satisfiability problem for Automatic Test Pattern Generation is implemented on IBM Quantum Experience.

II. TEST PATTERN GENERATION USING BOOLEAN SATISFIABILITY

The SAT problem, also known as the propositional satisfiability issue in software engineering, is the question of whether a translation exists that meets a given Boolean expression. Overall, it determines if the variables in a given boolean equation can be safely replaced by the values TRUE or FALSE, causing the formula to evaluate to TRUE. The condition is considered satisfiable if this is the case. In the absence of such an interpretation, the equation's limit is FALSE for all conceivable variable values, thus the statement is unsatisfiable.

Boolean Satisfiability Problem has been shown to be beneficial in a variety of applications, including VLSI circuit testing and machine learning. Its unique capacity to solve issues based on predefined restrictions solved a wide range of combinatorial problems, including university course scheduling and many others. Due to the tremendous increase in semiconductor demand, highly efficient and defect-free chips are in high demand, as these chips are at the heart of many critical issues[6]. The reliance on ATPG systems has grown in the last decade in order to design defect-free components, putting a strain on their ability to generate test patterns.

The whole problem of generating the boolean expression for single stuck at fault model in combinational circuit has been divided into two steps:

1) The first step is to emerge at a boolean expression which will be defining the set of test structures for a single stuck at fault model.

2) Further after generating the expression, an algorithm of boolean satisfiability will be run to get the values of the variables which will be the test pattern to detect that particular fault.

In order to demonstrate both the steps, the carry circuit of the full adder is considered. Figure -1 shows the carry part of the full adder circuit. The boolean expression for the carry circuit is

$$\text{Carry} = A * B + C\,(A \oplus B)$$

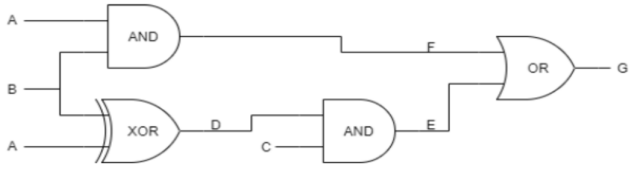


Fig. 1 Circuit diagram of Carry

In order to have a topological depiction of the circuit, DAG is being used for its representation. The hubs of the diagram are its inputs, its output, doorways and fan-out centres. Circuit lines also called as wires are represented as edges. Sources and the sinks of this DAG are the inputs and outputs of the circuit. The DAG of the carry circuit considered here as example is as shown in Figure -2

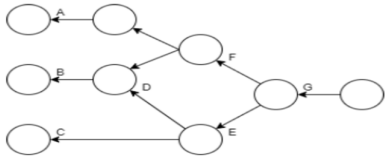


Fig. 2 Directed Acyclic Graph of Carry

Because the fanout points and gates are tagged with an expression, the characteristic formula of any circuit may be retrieved using the DAG by simply retaining the output node as the start point and walking the graph[7]. This is accomplished by combining the whole set of equations for all of the visited nodes. As a result, the carry circuit's characteristic formula is as follows:

(G+F) · (G+E) · (G+F+E) · (E+C) · (E+D) · (E+C+D) ·(D+A+B)·(D+A+B)·(D+A+B)·(D+A+B)·(F+A)·(F+B)·(F+A+B)

This characteristic formula is also the algebraic expression for the fault-free carry circuit. Figure -3 shows the fault-free circuit with characteristic formula of all its gates.

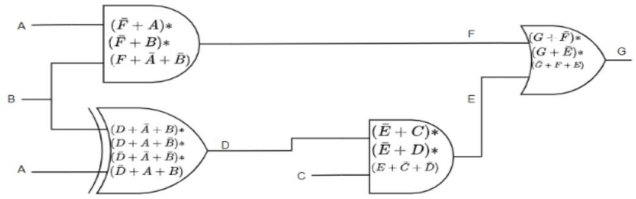


Fig. 3 Carry Circuit with labelled gates

Making a clone of this circuit and inserting two additional node locations where stuck at fault has influenced maintaining other nodes or variables intact can represent the faulty version of this circuit. The defective value will be generated at the fault location, but the other values will not change. Node E has been designated as the problem location in this case, with E's value remaining constant at 1. As a result, a new expression for the faulty circuit must be created. E' has been applied to the OR gate's input since the E node was not responding appropriately[8]. Because the input E solely affects the OR gate's output, the output has been modified to G'.Both the faulty and fault-free carry circuit have indistinguishable conduct apart from those nodes which are actually affected by those faults. The expression for the faulty is considered in the same way as compared to fault free. The characteristic boolean expression for the faulty circuit is:

$$(G'+F)\cdot(G'+E')\cdot(G'+F+E')\cdot(E')\cdot(F+A)\cdot(F+B)\cdot(F+A+B)$$

The faulty circuit of the given carry circuit is represented in Figure -4

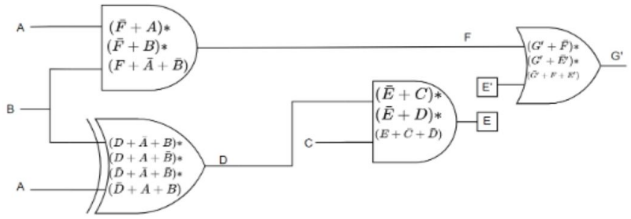


Fig. 4 Carry Circuit with E stuck at 1

To test the supplied flaw, a set of inputs that resulted in the faulty output being different from the fault-free output must be identified. Adding a third XOR gate between the two outputs can aid in determining which set of inputs both the faulty and fault-free outputs differ from. To obtain the final formula, the conjunction of the two expressions is used, along with the addition of the XOR formula. The following is the CNF formula, which takes into account both faulty and fault-free outputs:

(G'+F)·(G'+E')·(G'+F+E')·(E')·(F+A)·(F+B)·(F+A+B)· (G+F)·(G+E)·(G+F+E)·(E+C)·(E+D)·(E+C+D)· (D+A+B)·(D+A+B)·(D+A+B)·(D+A+B)·(G+G'+out)· (G+G'+out)·(G+G'+out)·(G+G'+out)

The final circuit with faulty and fault free output of the resulting expression is as shown in Figure:

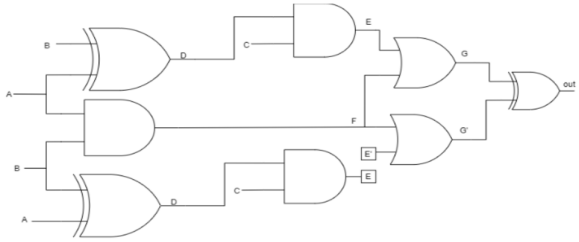


Fig. 5 Final carry circuit with XOR gate whose output is 1

A. Solving Boolean Satisfiability Problem

Using the foregoing strategy, an issue was transformed from one that was being resolved in exponential time depending on the number of its inputs in the worst case scenario to one that is now dependent on circuit variables. Regardless, the class of expressions generated for combinational circuits using the above method is noteworthy because it includes all three CNF formulas. This reality can be utilised to try to avoid every worst-case scenario possible.

Grover's search algorithm has been shown to be the most efficient approach for searching an unsorted database for an element. To discover the target element, this programme employs the amplitude amplification method. When contrasted to its classical equivalents, this technique can be used to find answers to unstructured problems at a quadratic speed. The Grover Instance is created using the oracle function. This instance is run and output is returned as the values of variables which satisfies the following considered boolean expression.

III. IMPLEMENTATION OF SATISFIABILITY PROBLEM

The Boolean Satisfiability problem for automatic test pattern generation is implemented on IBM Quantum Experience platform for finding the test pattern of ATPG systems and analysing the results.

A. IBM Quantum Experience

IBM Quantum Experience is a collective of IBM Quantum Composer and IBM Quantum Laboratories. It creates an online stage permitting general public and premium admittance given by IBM for cloud based quantum processing administrations and incorporates admittance to a bunch of model quantum processors from IBM. Currently, IBM has added about 20 service processors available in different geographical locations with differences in qubit counts. Clients cooperate with a quantum processor through quantum circuits created programmatically using Jupyter Notebooks[10]. These quantum circuits are compiled down to Open QASM for execution on real processors.

B. Solving Boolean Satisfiability on IBM Q

The actual implementation for the three-component SAT is done by utilizing the IBM Quantum Information Science Kit (QISKIT) using Python programming language and IBM Quantum Assembly (QASM).Fig-6 describes the flowchart of the algorithm. The steps of the algorithm are described as follows:

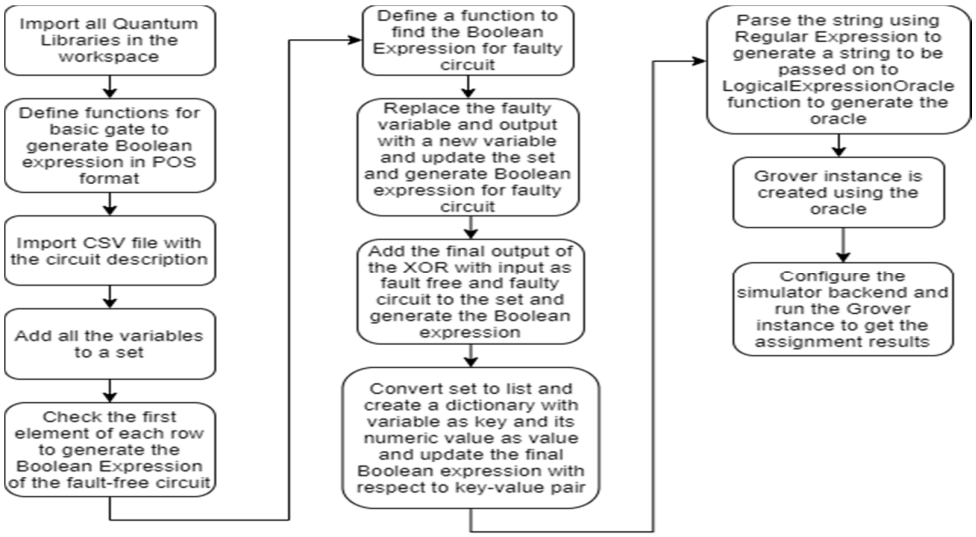


Fig. 6 Flowchart of the algorithm

1) *Import Libraries:* The first step is to import all the Quantum libraries in the workspace. Grover algorithm instance is imported from the qiskit.aqua.algorithms library.
2) *Expression for Basic Gates:* Once all the required libraries are imported, it is very important to define the logical expression of the basic gate in the CNF form.
3) *Importing Circuit Description File:* The circuit description of the considered carry circuit in Fig-1 has been specified in comma separated values (CSV) format and is imported into the workspace to generate the expression. Table-1 demonstrates the CSV file of the carry circuit. This CSV file is imported and each row is read. All the variables in the circuit are stored in a set which stores unique values.

Table 1: CSV File of Carry Circuit

Logical Gate	Input 1	Input 2	Output
XOR	A	B	D
AND	A	B	F
AND	C	D	E
OR	E	F	G

4) *Generate Boolean Expression:* After importing the circuit description file, CNF expression of the whole circuit is to be evaluated. The first element of each row is checked for the type of gate and function for that gate is called with inputs and outputs as its arguments. Empty string is considered and an expression for each row is appended to it to get the final boolean expression for the fault-free circuit.
5) *Faulty Circuit Expression:* This step is important as it defines a function to generate the boolean expression of the faulty circuit. It takes in two arguments the variable where the single stuck at fault is considered and value of fault whether it is single stuck at 0 or 1.The faulty variable is changed with a new variable and the output of gates affected due to this variable is also replaced with a new variable and the initial set considered is updated with these new variables. Also, the initial value of the row is checked for the type of gate and hence the expression for faulty circuit is generated which considers expression for the gates affected due to this fault.
6) *Generating Final Expression:* This step is the most important where both the output of a faulty and fault-free circuit is taken as the input of XOR gate and the final expression of the circuit is evaluated in conjunction with all the clauses present. The set with all the variables are converted to a dictionary with key as variables and value as numeric value.This final boolean expression is the final expression which will be used as an input to SAT problem. The function defined in the earlier step is also called with variable 'E' and its stuck at 1 is passed as arguments which indicates the fault model considered in this example i.e. E is stuck at 1.
7) *Generating the DIMACS CNF:* Once the final boolean expression with variables replaced with their numeric values is done, this step parses the string and generates another string in the DIMACS CNF which will act as input to Logical Expression Oracle to generate the oracle function for this particular SAT problem.
8) *Executing Grover's Algorithm:* This is the final step to solve the SAT problem. In this step, the DIMACS CNF string generated by the previous is passed to Logical Expression Oracle. The oracle is passed to the Grover function to create a Grover instance for the problem. The simulator background in the IBM server is initialized and the Grover instance is run on that simulator. The results are generated which contain the final Quantum Circuit with number of gates used and the final test pattern which satisfies the given SAT problem with highest probability. In this project, 'simulator mps' is used which is a 100 qubit simulator designed by IBM.

IV. RESULTS

Boolean Satisfiability problem for generation of test pattern in ATPG systems was implemented using Grover's Algorithm on IBM Quantum Experience platform and the results for the test pattern was obtained. The results of any quantum problem are observed in probability with the highest probability of the one with the correct assignment. For this paper, in which the carry circuit is considered with E variable stuck at 1 fault, the Grover circuit was generated with the oracle function. Fig-7 shows the grover circuit generated on the IBM Quantum composer.

For this particular case, the problem considered 10 input qubits and 40 extra ancilla bits are added for computation as it can be seen from the circuit. The run time for the whole algorithm on Quantum server is 77sec on the server which is of the same time complexity as considered in theoretical case i.e. $O(1.414^n)$. The time taken to solve this problem is less when considered with its classical counterparts to solve this problem. For this particular case with E stuck at 1 the test pattern with the highest probability is (A,B,C) = (1,0,0) which is the correct answer when solved the same problem using any other testing method. The quantum circuit can be transpiled to get the amount of different gates utilised to solve this problem with the depth of Toffoli gate. Table 2 describes the types of gates used with its count to solve this particular circuit which explains the amount of resources allocated for a particular problem to be solved on a Quantum computer.

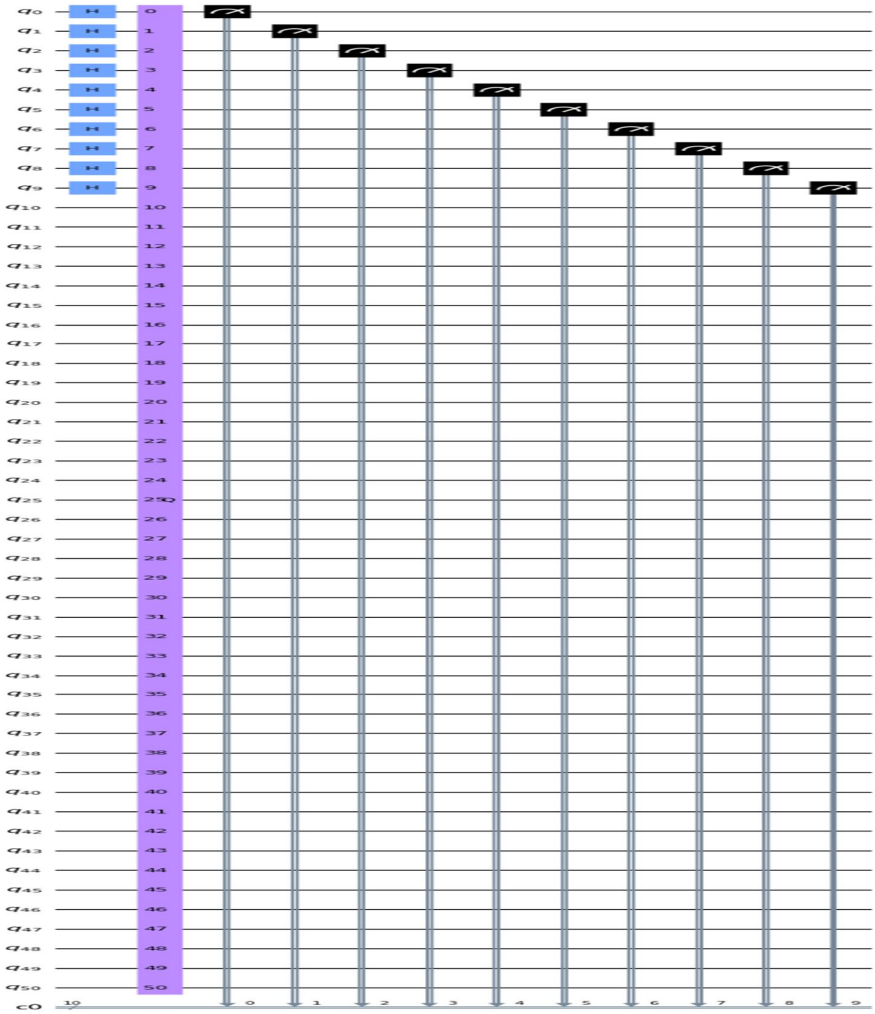


TABLE 2: Number of gates

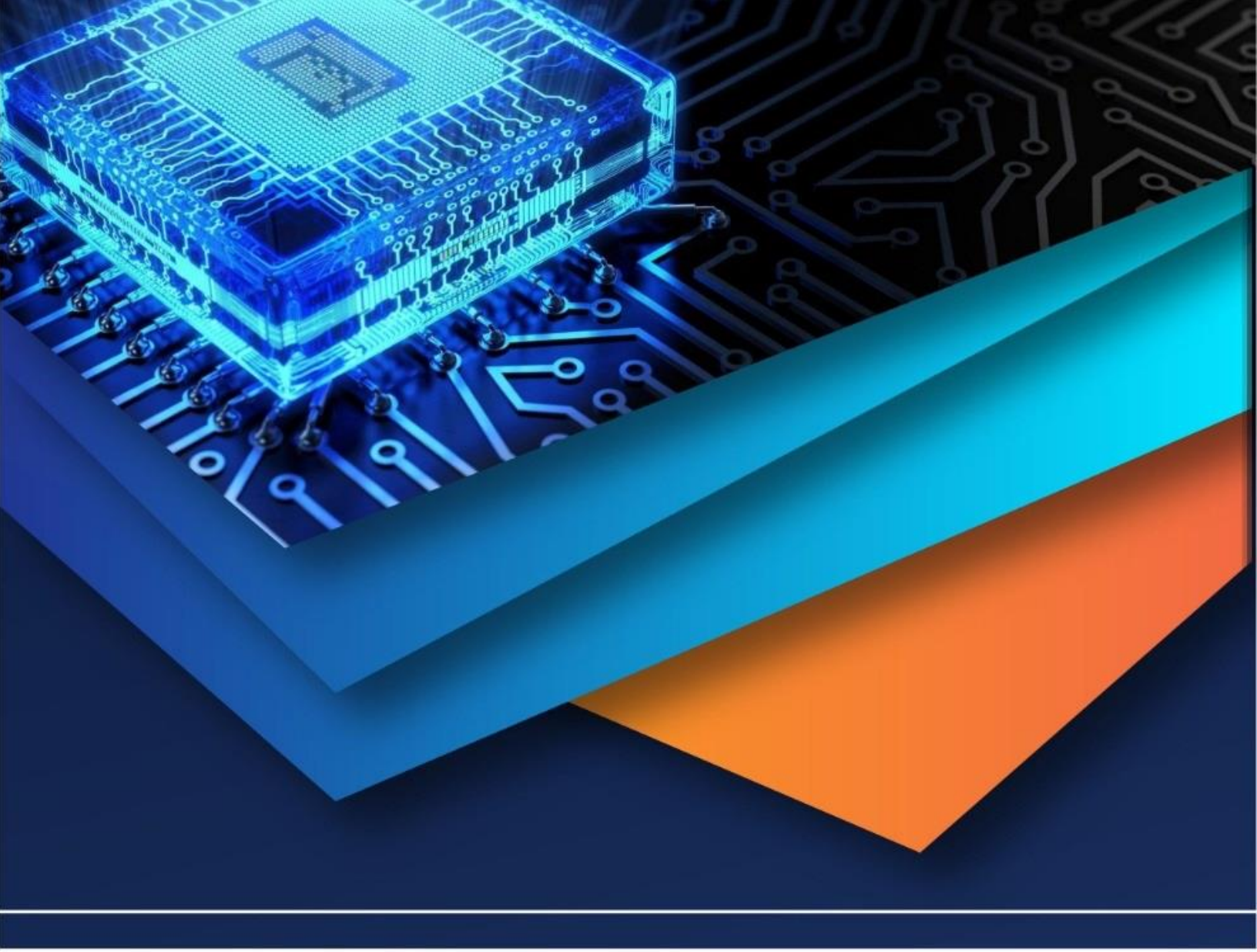
Quantum Gate	Count
cx (controlled x)	2022
u1,u2,u3 (unitary gates)	1695
t (toffoli)	123
x	53
h (hadamard gate)	2

V. CONCLUSIONS

The following paper proposed a method to apply boolean satisfiability method for generating efficient test patterns for single stuck at fault models and the satisfiability problem was solved by applying Grover's algorithm which provides a quadratic speedup for searching through unstructured databases when compared to its classical counterparts. The formula is extracted for the test set and grover's algorithm is run on it to get the correct assignment. Boolean satisfiability problems hence are very flexible and effective to be implemented on Quantum computers which in turn decreases the time complexity. The further situations can be solved for a large number of gates with more optimization in resources allocated when working on high level quantum computers with better optimization levels.

REFERENCES

[1] Z. Zilic and K. Radecka, "The role of super-fast transforms in speeding up quantumcomputations," inProceedings 32nd IEEE International Symposium on Multiple-ValuedLogic, IEEE, 2002, pp. 129–135
[2] R. P. Feynman, J. G. Hey, and R. W. Allen, Feynman Lectures on Computation. USA:Addison-Wesley Longman Publishing Co., Inc., 1998,isbn: 0201386283.
[3] D. Deutsch and R. Jozsa, "Rapid solution of problems by quantum computation," GBR,Tech. Rep., 1992.
[4] X. Zhou, D. W. Leung, and I. L. Chuang, "Methodology for quantum logic gate con-struction,"Physical Review A, vol. 62, no. 5, p. 052 316, 2000
[5] T. Humble, "Consumer applications of quantum computing: A promising approach forsecure computation, trusted data storage, and efficient applications,"IEEE ConsumerElectronics Magazine, vol. 7, no. 6, pp. 8–14, 2018.
[6] L. Gyongyosi, "Objective function estimation for solving optimization problems in gate-model quantum computers,"Scientific reports, vol. 10, no. 1, pp. 1–21, 2020
[7] M. Alam, A. Ash-Saki, and S. Ghosh, "Analysis of quantum approximate optimizational gorithm under realistic noise in superconducting qubits,"arXiv preprint arXiv:1907.09631,2019.
[8] T. Haener, M. Soeken, M. Roetteler, and K. M. Svore, "Quantum circuits for floating-point arithmetic," inInternational Conference on Reversible Computation, Springer,2018, pp. 162–174
[9] L. Vandersypen, M. Steffen, G. Breyta, C. Yannoni, M. Sherwood, and I. Chuang, "Ex-perimental realization of shor's quantum factoring algorithm using nuclear magneticresonance. nature, 414:883,"Nature, vol. 414, pp. 883–7, Dec. 2001.doi:10.1038/414883a
[10] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, "Quantum algorithms revisited,"Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences,vol. 454, Aug. 1997.doi:10.1098/rspa.1998.0164



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY