RV College of Engineering®
Autonomous institution affiliated to Visvesvaraya Technological University, Belagavi | Approved by AICTE, New Delhi
Go, change the world

Objective Function Estimation in Gate level Quantum computer

A Major Project Report
16EC81

Submitted by,

Kumar Shashank	**1RV17EC063**
Nishant Agrawal	**1RV17EC189**
Rahul J	**1RV17EC114**
Shashank R S	**1RV17EC140**

Under the guidance of

Dr. N. Ramavenkateswaran
Assistant Professor
Dept. of ECE
RV College of Engineering

In partial fulfillment of the requirements for the degree of

Bachelor of Engineering in

Electronics and Communication Engineering

2020-2021

RV College of Engineering®, Bengaluru

(*Autonomous institution affiliated to VTU, Belagavi*)

Department of Electronics and Communication Engineering

.

CERTIFICATE

Certified that the major project (16EC81)work titled ***Objective Function Estimation in Gate level Quantum computer*** is carried out by **Kumar Shashank** (**1RV17EC063**), **Nishant Agrawal** (**1RV17EC189**), **Rahul J** (**1RV17EC114**) and **Shashank R S** (**1RV17EC140**) who are bonafide students of RV College of Engineering, Bengaluru, in partial fulfillment of the requirements for the degree of **Bachelor of Engineering** in **Electronics and Communication Engineering** of the Visvesvaraya Technological University, Belagavi during the year 2020-2021. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the major project report deposited in the departmental library. The major project report has been approved as it satisfies the academic requirements in respect of major project work prescribed by the institution for the said degree.

Signature of Guide	Signature of Head of the Department	Signature of Principal
Dr. N. Ramavenkateswaran	Dr. K S Geetha	Dr. K. N. Subramanya

External Viva

Name of Examiners	Signature with Date
1.	
2.	

DECLARATION

We, **Kumar Shashank** , **Nishant Agrawal** , **Rahul J** and **Shashank R S** students of eighth semester B.E., Department of Electronics and Communication Engineering, RV College of Engineering, Bengaluru, hereby declare that the major project titled '**Objective Function Estimation in Gate level Quantum computer**' has been carried out by us and submitted in partial fulfilment for the award of degree of **Bachelor of Engineering** in **Electronics and Communication Engineering** during the year 2020-2021.

Further we declare that the content of the dissertation has not been submitted previously by anybody for the award of any degree or diploma to any other university.

We also declare that any Intellectual Property Rights generated out of this project carried out at RVCE will be the property of RV College of Engineering, Bengaluru and we will be one of the authors of the same.

Place: Bengaluru

Date:

Name Signature

1. Kumar Shashank(1RV17EC063)
2. Nishant Agrawal(1RV17EC189)
3. Rahul J(1RV17EC114)
4. Shashank R S(1RV17EC140)

ACKNOWLEDGEMENT

We are indebted to our guide, **Dr. N. Ramavenkateswaran**, Assistant Professor, RV College of Engineering . for the wholehearted support, suggestions and invaluable advice throughout our project work and also helped in the preparation of this thesis.

We also express our gratitude to our panel members **Prof. Sujata Priyambada Mishra**, Assistant Professor and **Dr. N. Ramavenkateswaran**, Assistant Professor, Department of Electronics and Communication Engineering for their valuable comments and suggestions during the phase evaluations.

Our sincere thanks to the project coordinators **Prof. Subrahmanya K N**, **Dr Nithin M** and **Dr Veena Devi** for their timely instructions and support in coordinating the project.

Our gratitude to **Prof. Narashimaraja P** for the organized latex template which made report writing easy and interesting.

Our sincere thanks to **Dr. K S Geetha**, Professor and Head, Department of Electronics and Communication Engineering, RVCE for the support and encouragement.

We express sincere gratitude to our beloved Principal, **Dr. K. N. Subramanya** for the appreciation towards this project work.

We thank all the teaching staff and technical staff of Electronics and Communication Engineering department, RVCE for their help.

Lastly, we take this opportunity to thank our family members and friends who provided all the backup support throughout the project work.

ABSTRACT

Quantum Computing (QC) is an interesting new field at the crossing point of Computer Science, Physics and Mathematics. The objective of QC is to channelize the extraordinary properties of Quantum Physics to perform computations that are past the capabilities of conventional classical circuits based on transistors. Gate level quantum computer plays a significant role in implementing various unique properties of quantum mechanics. Prominent applications of Quantum computation has motivated the consideration of generating efficient test pattern for ATPG systems. This work implements a quantum solver for Boolean satisfiability problem which helps in generation of test pattern for basic carry circuit.

For reliable computer systems, each of the components should be defect free. Any combinational circuit with the presence of single stuck at fault can be tested by applying a set of inputs which excite verifiable output response for that circuit. The outputs of that circuit will be different from the one desired if the faults exist. Automatic Test Pattern Generator (ATPG) is used to generate set of inputs to detect all single stuck at faults in the circuit. This project describes a method for generating test patterns using Boolean Satisfiability method. This process generates test pattern in two steps: First, a boolean formula is constructed expressing Boolean Difference between the fault-free and the faulty circuit. Second, Boolean Satisfiability Algorithm is applied to the formula from the previous step. The Boolean Satisfiability problem is solved using Grover's Algorithm.

The Boolean Satisfiability problem for Automatic Test Pattern Generation is implemented on IBM Quantum Experience an open source tool to access a set of IBM's prototype quantum processors. Circuit Description is uploaded as a comma-separated values (CSV) file. 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 run on IBM simulator. This method for produced excellent results on combinational circuits for test pattern generation with a quadratic speedup. Grover's Algorithm on this problem has a run time of $O(\sqrt{N})$.

CONTENTS

Abstract i

List of Figures iv

List of Tables vi

Abbreviations vii

1 Introduction to Quantum Computing 1

1.1 Introduction . . . 2

1.1.1 National and International overview of Quantum computing . . . 5

1.2 Motivation . . . 5

1.3 Problem statement . . . 6

1.4 Objectives . . . 6

1.5 Literature Review . . . 6

1.6 Brief Methodology of the project . . . 15

1.7 Assumptions made / Constraints of the project . . . 16

1.8 Organization of the report . . . 16

2 Theory and Fundamentals of Gate Level Quantum Computer 18

2.1 Introduction to QIP . . . 19

2.1.1 Quantum States and Qubits . . . 19

2.1.2 Superposition in Quantum Computing . . . 21

2.2 Quantum Gate model . . . 21

2.2.1 Quantum Circuits: . . . 22

2.2.2 Quantum logic Gates . . . 22

2.3 Quantum Algorithms . . . 24

2.3.1 Quantum Parallelism . . . 25

2.3.2 Deutsch-Jozsa Algorithm . . . 26

2.3.3 Grover's Algorithm . . . 28

2.3.4 Shor's Algorithm . . . 30

2.4 Challenges and Future of Quantum Computers . . . 31

3 Test Pattern Generation using Boolean Statisfiability **33**

3.1 Importance of ATPG systems 34

3.2 Boolean Satisfiability Problem 35

3.2.1 Introduction 35

3.2.2 Generation of test pattern using Boolean Satisfiability Method 36

3.3 Solving Boolean Satisfiability Problem 40

3.3.1 Existing Algorithms 40

3.3.2 Solving Satisfiability Problem using Grover's Algorithm 41

4 Implementation of Satisfiability Problem **43**

4.1 IBM Quantum Experience 44

4.2 Solving Boolean Satisfiability on IBM Q 44

5 Results & Discussions **53**

5.1 Output of the SAT problem 54

5.2 Output of the Grover's Algorithm 55

6 Conclusion and Future Scope **59**

6.1 Conclusion 60

6.2 Future Scope 61

6.3 Learning Outcomes of the Project 61

LIST OF FIGURES

1.1 Quantum computer [1] 2
1.2 Flow chart of design methodology 16

2.1 Quantum Computer developed by Google [17] 20
2.2 Bloch Sphere Representation of single qubits tates 20
2.3 Pauli X Logic Gate 22
2.4 Pauli Y Logic Gate 23
2.5 Pauli Z Logic Gate 23
2.6 Hadamard Logic Gate 24
2.7 Controlled NOT Logic Gate 25
2.8 Generic circuit of Deutsch Jozsa Algorithm [22] 27
2.9 Step 1 of Amplitude Amplification 29
2.10 Step 2 of Amplitude Amplification 29
2.11 Step 3 of Amplitude Amplification 29
2.12 Generic circuit of Grover Algorithm [25] 30
2.13 Flow Chart of Shor's Algorithm 31
2.14 Quantum Circuit of Shor's Algorithm 32

3.1 Circuit Diagram of Carry 36
3.2 Directed Acyclic Graph of Carry 37
3.3 CNF for basic gates 38
3.4 Carry Circuit with labelled gates 38
3.5 Carry Circuit with E stuck at 1 39
3.6 Final carry circuit with XOR gate whose output is 1‘ 40

4.1 Flow Chart of the algorithm 45

5.1 Evaluating boolean expression for carry circuit 54
5.2 CNF expression for 'E' stuck at 1 54
5.3 CNF expression for 'D' stuck at 1 55
5.4 CNF expression for 'A' stuck at 1 55
5.5 Test pattern for 'E' stuck at 1 fault 56

5.6 Number of reversible gates for 'E' stuck at 1 56
5.7 Test pattern for 'D' stuck at 1 fault 56
5.8 Number of reversible gates for 'D' stuck at 1 57
5.9 Test pattern for 'A' stuck at 1 fault 57
5.10 Number of reversible gates for 'A' stuck at 1 57

LIST OF TABLES

2.1 Truth Table for CNOT gate 24

4.1 CSV file of carry circuit 46

5.1 Summarised Results of the three cases 58

ABBREVIATIONS

ATPG Automatic Test Pattern Generator

CNF Conjunctive Normal Form

CNOT Controlled NOT

CSV comma-separated values

D-J Deutsch-Jozsa

DAG Directed Acyclic Graph

DIMACS The Center for Discrete Mathematics and Theoretical Computer Science

DPLL Davis–Putnam–Logemann–Loveland

EDA Electronic Design Automation

IBM International Business Machines Corporation

NMR Nuclear Magnetic Response

QASM Quantum Assembly

QC Quantum Computing

QFT Quantum Fourier Transform

QIP Quantum Information Processing

QISKIT Quantum Information Science Kit

QLC Quantum Logic Gates

QPE Quantum Phase Estimation

RSA Rivest-Shamir-Adleman

SAT Boolean Satisfiability Test

Chapter 1

Introduction to Quantum Computing

CHAPTER 1

INTRODUCTION TO QUANTUM COMPUTING

1.1 Introduction

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. Since then, has been developed in the fields of physics and computer science to overcome the main technical challenges of implementing reliable quantum computers on a large scale, although still has many challenges. Furthermore, Moore's law results in the IC feature size being less than 30nm, in which quantum effects begin to prevail. Another challenge faces is the development of quantum algorithms, which are currently very few. One of the most well-known quantum algorithms is Shor's factoring algorithm. Since the advantage of public key encryption technology used to protect data on computer networks is based on the difficulty of breaking down integers, the development of large-scale reliable quantum computers may allow crypt analysts to crack key ciphers public.

The design space of quantum circuits is huge. Shor's algorithm can be implemented using different quantum gate configurations. Furthermore, many other quantum error correction techniques have been developed, and each error correction algorithm has numerous parameters that can be adjusted depending on the design goals. Figure 1.1 shows a basic design of a Quantum computer at Google.

Figure 1.1: Quantum computer [1]

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. The concept of is called superposition: before measuring the value of the qubit, its value is indefinite and obeys the law of probability.There is a chance of measuring zero and a chance of measuring one, but either way, the quantum system will collapse, leaving 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, which is different from the classic N bit which can only represent one value at a time. In addition to superposition, quantum circuits also use entanglement, which is the phenomenon in quantum mechanics, in which two particles can be entangled. When one half of the interlaced pair is measured, the two-qubit system collapses, and measurements of the second half of the pair provide the same result as the first pair.

For to realize large-scale and reliable quantum computers, it will be necessary to evaluate various quantum error correction schemes and their parameter trade-offs.This will necessitate the development of design tools that allow designers to explore the design space by altering the quantum error correcting algorithm and its parameters. The analog quantum circuit is limited by the fact that the complexity of the simulation increases exponentially by the number of qubits.The quantum circuit that is being constructed will be displayed as part of the user interface. Designers will be able to use the mouse to manually add quantum gates. Designers will also be able to save the quantum circuit to a file using the application. This file can be used to simulate quantum circuits in a different application.

The development of viable quantum computers, accompanied by exponential growth in computing power, seems to revolutionize various industries and applications. The introduction of quantum computers is predicted to improve many computer applications with enormous data sets, and the majority of work done throughout the world is based on mathematical concepts from simulation to implementation.The Quantum computer has powerful computing power and is an ideal choice for solving the problems. This has created a world full of opportunities in almost every aspect of modern life.

1. Healthcare:

    - The Quantum computer will allow the simulation of larger molecules and allows researchers to model and simulate the interaction between drugs and more than 20,000 proteins encoded in the human genome to further promote pharmacological research.

- Quantum technology can be used to provide faster and more accurate diagnosis in a variety of applications.
- Quantum computers will enable therapists to run more simulations in less time, thereby helping to minimize radiation damage to healthy tissues.

2. Finance:

- Algorithmic trading, which uses complicated algorithms to automatically trigger stock trading depending on many market conditions, is one potential application of quantum technology.
- Quantum computers can greatly improve machine learning capabilities; greatly reduce the time required to train neural networks and increase the detection rate.

3. Marketing:

- Quantum Computer will be able to aggregate and analyze large amounts of consumer data from various sources.
- Big data analytics will allow government and business departments to precisely target individual consumers or voters based on their personal preferences and will help influence consumer spending and election results.

4. Meteorology:

- The use of quantum computers for machine learning will increase pattern recognition, making it easier to predict extreme weather events and potentially save thousands of lives each year.
- Climate scientists will also be able to create and analyse more complex climate models, giving them a better understanding of climate change and how to reduce its harmful effects.

5. Logistics:

- A wide range of sectors will be able to optimise workflows linked with transportation, logistics, and supply-chain management thanks to improved data analysis and modelling.
- Traffic management, fleet operations, air traffic control, freight and distribution could all benefit from the computation and recalculation of optimal routes.

1.1.1 National and International overview of Quantum computing

The unseen advantages of Quantum computer has motivated research in Quantum computer with various universities and companies are in the quest towards quantum supremacy. Government of India laid partnership with Finland involving IISER which will help in developing India's first quantum computer. They have also allocated huge amount of money towards Quantum computer and DST has launched Quest programme to lay the groundwork for developing of quantum computers. The motive is to initially design a five qubit quantum computer and scale it to 50 qubits. Indian Space Research Organisation (ISRO) is working towards implementing secure communication over open communication channel using quantum cryptography and have tested their algorithm for the initial use for a short distance. Pharmaceutical companies are investing in Quantum computation to accelerate the drug discovery process.

Superpower countries like China, United States have been developing their efficient quantum operating system to exploit the unique properties to speed up the computation. International Business Machines Corporation (IBM) has over 20 functional quantum computer over various locations which are open source and can be accessed over IBM cloud. IBM has partnered with Mercedes Benz to develop technologies to simulate batteries to make Electric Vehicles more efficient. Exxonmobil has also partnered with IBM to solve complex energy challenges which involve model maritime inventory routing for efficient shipment of LNG through ships over various locations. Google simulated a calculation using quantum computer which took just 200 seconds as compared to classical super computers which would take around 1000 years. Many researchers at college are working on simulating the quantum effect on nanoscale devices.

1.2 Motivation

1. Quantum computing applications include secure computing, reliable data storage and exponential acceleration.

2. Quantum computers are believed to greatly help solve optimization-related problems, which play a key role in everything from national defense to financial transactions.

3. The two most important application areas of quantum computing are

    - Quantum detection and measurement: Utilizing the extreme sensitivity of qubits to the environment.

- Quantum networking and communication: This can lead to revolutionary ways of processing and sharing information

1.3 Problem statement

Quantum circuit design for objective function maximization in gate-model Quantum Computers. This quantum circuit is applied to efficiently generate test patterns for single stuck at fault model for Automatic Test Pattern generation systems.

1.4 Objectives

The objectives of the project are

1. To analyze different Quantum Logic Circuits and Quantum Algorithms.
2. To design the Quantum circuits using Quantum Algorithms for a given objective function.
3. To implement the designed Quantum Circuit on IBM Quantum Experience and run it on IBM back end and analyse the results with respect to classical computations.

1.5 Literature Review

Paper[1] has proposed an efficient method to optimize the parameter values of the structure and the quantum circuit at the same time with very little computational overhead. The performance of the shallow circuit optimized by the structure is significantly better than that of the circuit using only parameter updates, which makes this method particularly suitable for noise medium-scale quantum computers. The method of optimizing is demonstrated using the variable quantum eigen-solver to find the ground state of lithium hydride and Heisenberg model in the simulation, and find the ground state of hydrogen in the IBM Melbourne quantum computer. In the case of circuit optimization, Rotosolve can be generalized to find K rotation angles and rotation angle minimizers at the same time, but 3K circuits need to be evaluated. Although this is an exponential cost, for small K, the method is computationally feasible and may provide advantages. This method belongs to the algorithm class and is called coordinate block minimization. In the case of circuit structure optimization, Rotoselect can be generalized to start from scratch instead of gradually increasing the growth circuit from a random initial structure. This is similar in spirit to Adapt-VQE , but the advantage is that each new gate effectively optimizes in a closed way, rather than using gradient-based optimization. Also, it can effectively eliminate gate by evaluating whether the gate contributes to the solution. Another

interesting extension of is the use of Rotoselect to learn the best connection design for tangled doors. For example, consider an ion computer that can implement a connecting layer of 4,444 Mølmer-Sørensen gates. This choice provides greater expressive power in shallow circuits, but the potential slow clock speed of these entangled gates must be weighed against potential gate errors. The algorithm can find this best position. Since n qubits are used, it can evaluate $\frac{n(n-1)}{2}$ options for non-directional two qubit gates in a layer, so this method is also valid.

Paper[2] discusses that Quantum computers provide valuable resources for solving computational problems. The maximization of the objective function of the computation problem is the key problem in the gate model quantum computer. Estimating the objective function is an expensive process that requires several rounds of quantum calculations and measurements. Here, it defines an objective function estimation method for arbitrary computational problems in gate model quantum computers. The proposed solution significantly reduces the cost of estimating the objective function and provides an optimal estimate of the state of the quantum computer to solve the optimization problem. The gate model quantum computer provides an achievable architecture for experimental quantum computing. Here, the problem of estimating the objective function in the gate model quantum computer is studied. The proposed framework uses measurement results and improves the accuracy of objective function estimation and maximization through calculation steps. This method reduces the costs associated with the physical layer, such as quantum state preparation, quantum calculation rounds, and measurement rounds. The expansion process of the objective function, the segmentation algorithm using quantum computer unit gate parameters, and the machine learning unit used to predict the state of the system. This result is particularly suitable for optimizing the performance of experimental gate model quantum computers and quantum internet devices in the near future.

This paper[3] discusses the short-term quantum computers, operations are performed by unit quantum gates. The precise and stable working mechanism of quantum gates is essential for the execution of any complex quantum calculations. Here, a method for unsupervised control of quantum gates in quantum computers in a short period of time is defined. An unstable quantum gate tensor product structure that cannot be controlled by control theory is modeled. It is shown that if the quantum gate forms an entangled gate structure, the unstable quantum gate will become controllable through machine learning methods. Here, a method for unsupervised control of entangled quantum gates in gate model quantum computers and short-term quantum devices is defined. The framework uses control theory terms to describe control problems. The

system model uses a quantum bus scheme, which associates the grid output with the auxiliary probe beam. The state of the probe is measured to provide input to the post-processing unit, and then input to the machine learning control. The tangled door structure is achieved by a second metering block that entangles the door exits, resulting in a tangled door structure. It is proved that if quantum gates are entangled, then unstable gates can be controlled by stable quantum gates. However, if the door forms a product system, the random oscillation of the unstable door cannot be controlled according to the control theory. This solution stabilizes the quantum gate structure by deriving an optimal control function that minimizes a specific cost function. This framework provides an achievable solution for experimental quantum computing and short-term quantum computer architecture on quantum networks.

Here[4] the gate model quantum computer provides an experimentally achievable architecture for short-term quantum computing. In order to design a simplified quantum circuit that can simulate a highly complex quantum reference circuit, the number of input quantum states, the unit operation of the quantum circuit, and the number of output measurement rounds must be optimized. In addition to optimizing the physical design of the hardware layer, quantum computers should also solve difficult problems very effectively. In order to produce the desired output system, the specific objective function associated with the computational problem introduced into the quantum computer must be maximized. The simplified gate structure should be able to generate the maximum value of the objective function. These parallel requirements must be met at the same time, making optimization difficult. A method for designing quantum circuits for gate model quantum computers and define the quantum triple annealing minimization (QTAM) algorithm. The goal of QTAM is to determine the best simplified topology for the quantum circuit in the hardware layer while maximizing the objective function of any computational problem. The algorithm and method presented in this document provide a framework for the design of quantum circuits for quantum computers short-term door model. As purpose is to define a plan for current and future quantum computers, the developed algorithm and method were adjusted for any quantum dimension system and any quantum hardware limitation. The results through the architecture of the quantum computer of the GATE model. However, for the flexibility of the scheme, any implementation and input restrictions can be integrated into the minimization of quantum circuits. The target function that is going to be maximized in this way can also be arbitrarily selected. This allows flexible implementation to solve any computational problem for experimental quantum computers with any hardware restriction and development restric-

tions.

Paper[5] discusses about Machine learning has developed rapidly, has made many breakthroughs in theory, and has been widely used in various fields. As an important part of machine learning, optimization has attracted widespread attention from researchers. With the exponential growth of data volume and the increase of model complexity, optimization methods in machine learning are facing more and more challenges. In order to solve optimization problems or improve optimization methods in machine learning, many works have been proposed one after another. It is very important to review the system afterwards and summarize the optimization methods from the perspective of machine learning, which can provide guidance for optimization development and machine learning research. In this article, first the optimization problem in machine learning is described. Then the principle and progress of commonly used optimization methods are presented. Finally, it is explored and proposed some challenges and unsolved problems to optimize machine learning. This document introduces and summarizes optimization methods from the perspective of machine learning, and studies the application of methods in various machine learning fields. First, it describes the theoretical basis of the optimization method based on the first-order, high-order and derivative-free aspects, and introduce the research progress in recent years. Then, it describes the application of the optimization method in different machine learning scenarios and the methods to improve its performance. Finally, it is discussed that some challenges and unresolved problems in the machine learning optimization method.

The author[6] provides a complete review of the numerical simulation and optimization of shot peening found in the existing literature in the last 10 years. The review found that the developed numerical models combining the finite element and the discrete element are becoming increasingly mature and show their advantages by combining flow behavior and projectile randomness. Great importance must be attached to the constitutive equation of the target material. It is recommended to incorporate its sensitivity of strain rate, cyclical behavior and Bauschinger effect in the numerical model of material at the same time, since considering one of them in isolation can lead to an unreliable residual stress distribution. In addition, the hardening of the material is a key advantage of shot blasting. However, whether in simulation or optimization, it has not received the attention of existing research. The study found that strength and coverage are two key control parameters, and it is recommended to use them as constraints to optimize shot peening. Finally, this work also found that recently developed heuristic algorithms have

shown their advantages and can find the best combination of shot peening parameters. In the near future, the synergy of combining these algorithms with approximate models is expected to attract more attention from researchers.

Paper[7] discusses the Approximate Quantum Optimization Algorithm (QAOA) is a promising classical quantum hybrid technology for solving the combined optimization problem in noisy short-term gate-based quantum devices. In QAOA, the target is a function of the quantum state, and the quantum state itself is a function of the gate parameters of the multilevel parameterized quantum circuit (PQC). The classical optimization procedure changes the continuous gate parameter to generate a distribution that has significant support for the best solution. Even at the lowest circuit depth, QAOA can still provide demonstrably important performance guarantees, and is expected to increase by with the depth of the circuit. However, the existing analysis does not take into account the non-ideality in qubit quality, that is, the short lifespan and imperfect operation of gate in real quantum hardware. In this article, the impact of various noise sources on QAOA performance in simulations and in real IBM quantum computers is studied. The analysis shows that the optimal number of stages (p-value) of any instance of QAOA is limited by the noise characteristics of the target hardware, which is different from the current view, that is, QAOA of greater depth will provide better monotonous performance for a Issue given compared to shallow deployments. The performance of QAOA-MaxCut using the real noise properties of superconducting qubits is analyzed. The results shows that for any single-instance QAOA optimization process, the best p-value will be limited by the qubit quality metric of any target hardware. For the test case considered in this work, the maximum value of the best p is 2. For most actual size problems with existing error rates, any value higher than beyond p = 1 does not guarantee performance improvement. Although the optimal p-value limit is instinctive due to the limited coherence time, the similar limit due to gate error is important. The results show that even if it can achieve qubits with an infinite coherence time of the optimal depth of any QAOA single instance optimization process may be limited by the gate error rate of the hardware destination.

Paper[8] discusses the needs of the business, finance, security and logistics sectors are driving the continued growth of IT. Current research focuses on using destructive methods to provide new computing functions. Quantum computing represents a promising strategy that can provide new functions in secure computing, reliable data storage, and efficient applications. This article focuses on the growing understanding that quantum computing will affect a variety of

consumer concerns and applications.

The author in paper[9] tells about Quantum algorithms that can solve practical problems in quantum chemistry, materials science, and matrix inversion generally involve a large number of arithmetic operations that act on the superposition of inputs. These must be compiled into a collection of fault-tolerant low-level operations, with the compiler's purpose being to grow closer to the Pareto optimal frontier throughout the conversion process between the number of qubits required and the depth of the resulting circuit. A quantum circuit for floating point addition and multiplication, and found that it uses two very different methods is provided. The first method is, which uses synthesis tools to automatically generate circuits from the classic Verilog. The second method is to manually generate and optimize these circuits. The two methods are compared and provide evidence that, at least for typical scientific applications, floating point arithmetic is a viable candidate for quantum calculations. The combination of and manual circuit optimization relaxes the requirements and can greatly save the width and size of the circuit, so that floating point arithmetic is more practical for future quantum device use. Also, since the cost of multiplying with fixed point numbers is very similar to floating point multiplication, using floating point arithmetic in typical scientific applications will cause to generate only 2-3 times the overhead in size of the KQ circuit.The goal function employed in the classical computational optimization process is substantially different from the objective function utilised in quantum computing, which is one of the reasons for the enormous discrepancy between the two methods.

Paper[10] describes the gate model quantum computer with too many qubits is about to be simulated by the available classical computer. They have proposed a strategy to program these devices without bug fixes or compilation. This means that the number of logical qubits is the same as the number of qubits in the device. The hardware determines which pair of qubits can be addressed by a single operator. The goal is to establish a quantum state to solve computational problems, such as maximizing the combined objective function or minimizing the Hamiltonian. These problems may not naturally fit the physical design of qubits. The algorithm uses parameterized unit sequences in the qubit design, and quantum states are generated based on these parameters. The measurement of the target guides the selection of new parameters. The goal is to move the target function upward. For example, if considered finding the approximate solution of MaxCut on 3 regular charts, and the hardware is physical qubits arranged in a rectangular grid. It is shown that in all sufficiently large cases, the lowest

depth version of the quantum approximation optimization algorithm will achieve an approximation ratio of at least 0.5293 of which exceeds random guesses. It turns on the algorithm to have different parameters to accommodate each rotation of the qubit X and each ZZ interaction related to the closest neighbor interaction in the grid. Small numerical experiments show that the classic envelope algorithm can be used to find the parameters, which can be found on the grid to optimize the objective function with different connectivity. They have discussed strategies for finding suitable parameters, but still have not provided evidence that the proposed method can beat the best classical algorithm. Ultimately, the strength of this method will be determined by running on actual hardware.

The author in paper[11] discusses about the FONMAN architecture of the classic computer comprises a central processing unit, instruction and memory retention data. It is demonstrated that the quantum random access memory integrated into the chip and demonstrate central processing unit are integrated into the chip. Trying quantum machine by executing the code that runs two Queen Superconductors coupled through two quantum buses, two quantum memories and two records to zero. Two important algorithms are shown for quantum calculus, the Fourier quantum transform, 66% fidelity and three-elbows toffee class or phase doors have a phase drawer of 98%. The results show a potentially executable approach to factoring fractions, especially in combination with a particularly long consistency of the quit particularly long, and implements a simple quantum error correction signal. Quantum processors based on nuclear magnetic resonance, captured ions and semiconductor devices are used to perform Shor's quantum decomposition algorithm and quantum error correction. The quantum operations on which these algorithms are based include two-qubit gates, Fourier quantum transform and three-qubit Tofli gates. In addition to quantum processors, the second key element for quantum machines is quantum memory. Circuit superconductor has reached multiple milestones, including demonstrations of two-qubit gates and advanced control of qubits and photon quantum states. Here a superconducting integrated circuit that combines a processor that performs a quantum Fourier transform and a Toffoli-like three-bit OR gate, and has a memory and a clear register in a single device is demonstrated. The combination of quantum central processing unit (quCPU) and quantum random access memory (quRAM) constitutes two key elements of the classical von Neumann architecture and defines the quantum von Neumann architecture.

Paper[12] discusses that in recent years, due to the possible application of quantum computing to the difficult problems of classical calculus, it has received great attention. Although there are

experimental problems with realizing quantum devices, theoretical physicists still try to design some implementations for quantum algorithms. Here some explicit schemes are introduced for performing basic arithmetic operations. Analysis shows how to build circuits for quantum addition and multiplication. It should be noted that in both cases, the input is the vector on which the calculation is based, and therefore the output vector. An important result is that although the process uses the superposition of the sum of state and phase, the final state does not have the typical ambiguity of quantum mechanics. The observable value measurement allows to obtain one of the eigenvectors of the Hermitian operator associated with it. In the example, the output is already calculating the basis vector, so the measurement result is deterministic. The circuit implemented by can be used to construct a more complex schematic that can perform more complex operations. As shown in the upcoming article, the circuit in question is the basis of the ALU of early quantum CPUs. In the process of system evolution, the problems of decoherence and control was completely ignored. The experimental work shows how difficult it is to manage real qubits. As a result, it is difficult to obtain circuit through the prior art method.
Paper[13] tells about how Quantum computing provides great speed when performing tasks, such as data encryption and searching. Quantum algorithms can be modeled using classical computing devices, but classical computer simulations can not be addressed efficiently with the parallel processing that exists in the quantum algorithm. Quantum circuit models of the quantum algorithm are sufficient to describe known quantum algorithms. Simulators between quantum circuits and digital circuits are used to design an emulator of quantum algorithms in FPGA and allow efficient experiments with new quantum algorithms. This document concentrates important technologies in the modeling of quantum circuits, including the implementation of entanglement and probability calculations, as well as important issues in the required computing accuracy. Here the problems in the design and operation of the FPGA-based quantum circuit simulator, and developed the platform for the development of quantum circuits is introduced. The simulator allows the construction of fairly complex quantum circuits from the component library in a simple way. At the same time, simulates the parallelism existing in the quantum computer by constructing a parallel evolution path for each qubit in the FPGA. FPGA simulation has the advantages of because it is difficult to effectively simulate the parallel evolution of quantum systems in software. The simulator is also extensible and has the potential to simulate complex quantum circuits. The simulator can also incorporate other quantum computing concepts, such as quantum error correction, fault-tolerant quantum computing

and quantum measurement technology. These are particularly useful in the development of practical systems for quantum computers. Other uses of the quantum simulator may be the analysis, optimization and approximation of the quantum Fourier transform, which is essential for a better quantum algorithm. It is planned further to explore the development of the reversible quantum gate library and the dedicated architecture for simulating quantum algorithms. Finally, a plan to use this simulator to study and optimize the quantum measurement algorithm, which is currently not practical in software simulation.

This paper[14] proposes a general method for building fault-tolerant quantum logic gates with simple primitives, which are simulations of quantum teleportation. This technique extends previous results based on traditional quantum teleportation, and leads to direct system construction of many fault-tolerant encoding operations. This technology can also be applied to the construction of remote quantum operations that cannot be executed directly. Here it is proposed that a system technology uses the original 1-bit teleportation scheme to construct various quantum operations. Such a solution reduces the problem of constructing a quantum logic gate to prepare the auxiliary state generated by the same gate applied to a known state. For two types of applications, the practicality of this technology is particularly obvious: fault-tolerant quantum computing and remote quantum computing, the $\pi/8$ gate, phase control and Toffoli and the construction of the remote junction have been proved. With the recursive application of one-bit recursive scheme, it can also build infinite layers of fault-tolerant gates. This two-bit transfer scheme allows all C3 gates to be fault-tolerantly transferred fault-tolerant, and all Ck gates are transferred through the recursive application of this scheme. However, for a bit transfer, it can only provide sufficient conditions for the gate in C3 to be transferable, that is, any C3 gate node that can write into the product of the C3 gate and the single switched C3 gate. I don't know if this includes the C3 gate. It is difficult to describe a set of accurate one-bit transmission C3 gates. The difficulty is caused by the need for C2C1 gates in a one-bit transmission circuit. This kind of gate C2C1 can be conjugated from C2 through gate C3, so cannot be directly executed with fault tolerance. Due to the current lack of understanding of the general structure and nature of the Ck gate, the difference between the maximum transmission capacity of 1444 and 2 bits mode is still an interesting and difficult open question. However, as already shown, in the construction of quantum logic gates, one-bit transmission can provide a simpler protocol than two-bit transmission. This is because one-bit teleportation requires only Z projection measurement and as many auxiliary qubits as the state to be converted however,

two-bit transmission requires Bell measurement, and the number of auxiliary qubits is two of the original state.

Paper [15] introduces role of spectroscopic methods in the development of the most impressive quantum algorithms. Although the fast transformation algorithm reduces the number of operations required for the transformation from O(2^{2n}) to O(n^{2n}), in contrast, quantum transformation is super fast. The quantum Fourier transform can be performed in O(n^2) time, while the specific case of Walsh-Hadamard and Chrestenson transform only requires O(n) operations. They have used Chrestenson gates to derive the quantum Fourier transform on non-binary quantum numbers, which can be used as the basic building block of quantum transforms. The role of quantum transformation in the design of efficient quantum algorithms is reviewed. The circuit for fast quantum transformation is, for example, multi-domain transformation. More work can be done on the design of quantum transformation circuits in binary and ternary quantum gates. It is worth studying whether the same acceleration technology is applicable to the ternary quantum circuit structure. Finally, as quantum machines are physically implemented to verify the Shor and Grover algorithms, and as some current promising technologies may become a reality, it is necessary to reconsider quantum transformation circuits in consideration of this implementation.

Among them, the most expensive resource is time, and the bits are really free. Therefore, the circuits produced by the optimized process designed to minimize the cost function of classical calculations are highly parallel, but also require more bits. On the other hand, in quantum computing, the depth and width of the circuit of are valuable resources. This makes it difficult to enter parallelism, and optimizing the program will generate parallelism with very different functions and circuits with minus digits. Although manual optimization of the circuit requires fewer qubits and T gates, it is very likely that can still use method in the automatic synthesis method to further optimize some subroutines. Also the interaction between the different components of the handwriting circuit can benefit from such a process.

1.6 Brief Methodology of the project

Figure 1.2 shows the basic flowchart of the methodology of the project.

- Initially the analysis of different quantum gates will be done based on their performance and computation complexity for different operations.
- Then Quantum circuits will be analysed based given quantum logic gates.

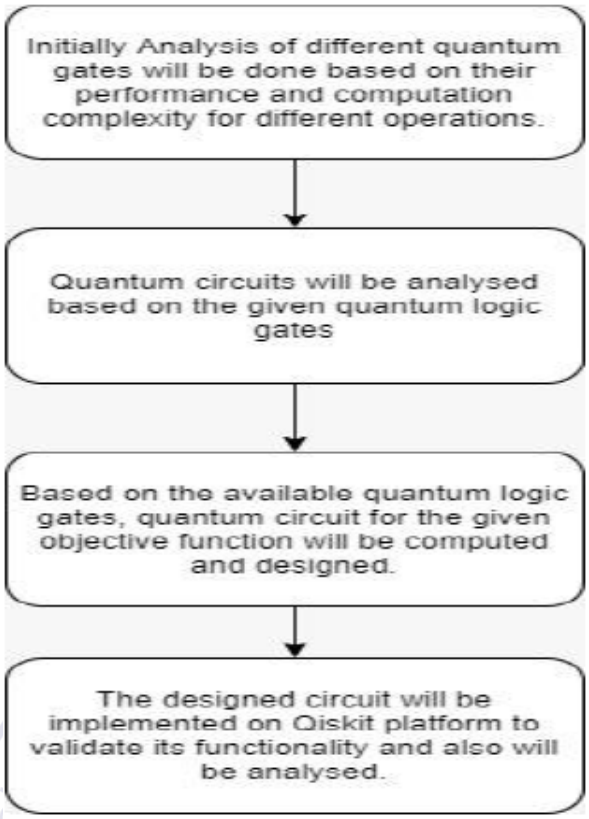


Figure 1.2: Flow chart of design methodology

- Depending on the available quantum logic gates, quantum circuit for the given objective function will be computed and designed.
- At last the designed circuit will be implemented on the Qiskit platform to validate and analyse its functionality.

1.7 Assumptions made / Constraints of the project

- The fault modelling scheme used is single stuck at fault.
- The analysis is done only for one primary output.
- The circuit with less number of gates is been used.

1.8 Organization of the report

This report is organized as follows.

- Chapter 2 discusses the fundamentals of gate level Quantum computer.
- Chapter 3 discusses about the test pattern generation using boolean satisfiability .

- Chapter 4 discusses implementation of the Boolean satisfiability.
- Chapter 5 discusses simulation results of the algorithm.
- Chapter 6 discusses conclusion and future scope.

.

Chapter 2

Theory and Fundamentals of Gate Level Quantum Computer

CHAPTER 2

THEORY AND FUNDAMENTALS OF GATE LEVEL QUANTUM COMPUTER

Quantum Information Processing (QIP) makes use of qubits, which are two-state quantum-mechanical frameworks that can be in superposition of both states at the same time for computing purposes. This chapter discusses the fundamentals of Quantum Computation along with various Quantum Logic Gates (QLC). Quantum Algorithms and comparison with its classical counterparts has also been discussed in this chapter. This chapter also describes the challenges faced by present day Quantum Computers and its future.

2.1 Introduction to QIP

Quantum Information is the information of the state of a Quantum systems which is an extension and completion of classical information. This Quantum information is encoded, transformed and decoded using QIP proposed by Feynman and Deutsch [16]. Quantum theory is actually probability theory with negative numbers. Traditional computer science, quantum mechanics, and quantum information theory are all consolidated into Quantum Computing (QC) . Physical realizations of Quantum Computer will follow the concepts of quantum physics and will be based on Quantum-Dots, nuclear magnetic resonance, superconducting junction and ion traps. Quantum Computer has found its usage in many applications due to its performance gains arising from Quantum Parallelism. This ability can be exploited to factorise bigger numbers or search large databases. These tasks are computationally expensive on classical computers. Quantum Algorithms like Shor's Algorithm and Grover's Algorithm can perform these tasks at a very high speedup. These quantum algorithms are based on quantum mechanics and hence its development is hard as compared to classical algorithms. Quantum Computers are no doubt better than classical computers at some problems but may not replace it. Hence, Quantum computers will be controlled classically.

2.1.1 Quantum States and Qubits

A quantum bit, also known as a qubit, is a basic unit of quantum information that is the quantum equivalent of classical binary bits. Qubits are two-state quantum mechanical systems which displays the eccentricity of quantum mechanics. Binary Digits represented as 0 or 1 are implemented as one of the two levels of low DC Voltage. Bits can exist in either one of

Figure 2.1: Quantum Computer developed by Google [17]

the states at a time but a qubit can also be in coherent superposition of both. But once the measurement is done on this superposition state it collapses to one of its states with higher probability.

Dirac Notation is used to describe coherent quantum states represented by a linear superposition of two orthonormal basis states. They are also called as "bra-ket" notation written as $|0\rangle$ and $|1\rangle$. These also form orthonormal basis states called as computational basis.

$$|0\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} and \; |1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{2.1}$$

These single qubit states can be represented using the Bloch sphere as shown in Figure 2.2.

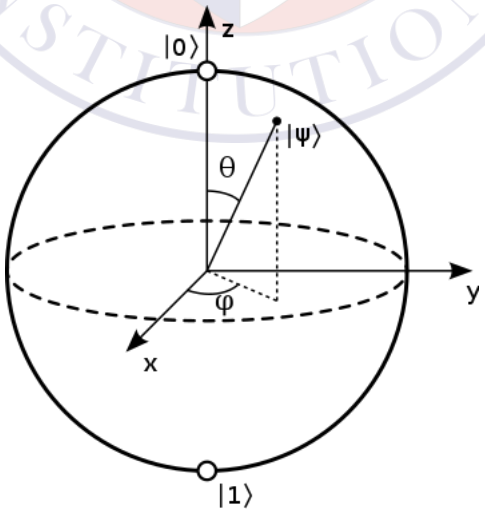


Figure 2.2: Bloch Sphere Representation of single qubits tates

A single qubit is represented in the form of vector representation as shown in equation 2.2:

$$| a\rangle = \nu_0 | 0\rangle + \nu_1 | a\rangle \longrightarrow \begin{bmatrix} \nu_0 \\ \nu_1 \end{bmatrix} \tag{2.2}$$

ν_0 and ν_1 are called the complex probability amplitudes of the qubit. These values help in determining the probability of measuring a 0 or 1 when the state of the qubit is measured.

2.1.2 Superposition in Quantum Computing

The ability of quantum system to be in superposition stage has proved to be very advantageous in quantum computing. Quantum superposition is the fundamental principle of quantum mechanics stating that two or more quantum states can be added to form another quantum state and vice versa. It is much like wave principle in classical physics. It states that if $| a\rangle$ and $| b\rangle$ are two states of a Quantum system , then superposition represented as $\alpha | a\rangle + \beta | b\rangle$ will also be an allowed state of a quantum system with

$$| \alpha |^2 + | \beta |^2 = 1 \tag{2.3}$$

It can also be understood as the probabilities of measuring either state should sum up to 1.[18] Bell state is the most common superposition state to display equal superposition between two qubits represented as:

$$\frac{1}{\sqrt{2}}(| 00\rangle + | 11\rangle) \tag{2.4}$$

2.2 Quantum Gate model

To conduct calculations, the quantum gate model employs a succession of quantum logic gates. The procedure starts with a state that is initialised to the computational basis, i.e. '0' or '1', then applies a sequence of quantum logic gates and reads out the outcome by measuring the qubits on the computation basis. Initialization, universal gates, and readout are examples of qubit manipulations performed at the lowest hardware levels. The error rates for qubit manipulations must be below threshold error rates, which vary depending on the error correction methodology utilised, for fault-tolerant quantum processing. Classically difficult computer problems, such as factoring huge numbers, are examples of potential applications.

The number of steps necessary to solve some linear algebra problems using parallel quantum circuits is independent of the problem size, according to a theoretical explanation, but

the no of steps required for conventional circuits grows logarithmically with problem size has just been published. Database search, portfolio optimization, machine learning, and combinatorial optimization are some of the other applications. Quantum circuits, have quantum correlations, which give them a quantum advantage. Classical supercomputers now available cannot simulate quantum computers with more than 50 sufficiently coherent qubits.

2.2.1 Quantum Circuits:

A quantum circuit is a computing mechanism that combines real-time classical processing with coherent quantum operations on quantum-data like qubits. It's a system of quantum gates, measurements, and resets that can fed data from real-time classical processing after conditioning together. Quantum gates are unitary gates, which implies the number of qubits in the inputs and outputs must be the same, and unlike classical gates, unitary gates are always reversible.

2.2.2 Quantum logic Gates

A quantum logic gate is a fundamental quantum circuit that works with a modest number of qubits. Followings are the types of Quantum Logic Gates:

1. **Pauli X gate** - This gate only affects a single qubit. In terms of standard bases, it's the quantum counterpart of the NOT gate. It converts $|0\rangle$ to $|1\rangle$ and $|1\rangle$ to $|0\rangle$. Pauli X matrix is usually represented as:

$$X = \sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2.5}$$

The Pauli X Logic Gate and its similarity with NOT gate is as shown in Figure 2.3

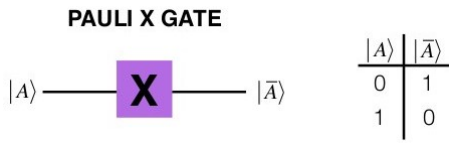


Figure 2.3: Pauli X Logic Gate

2. **Pauli Y gate** - This gate only affects one qubit. It translates $|0\rangle$ to $|i\rangle$ and $|1\rangle$ to

$|-i\rangle$.Pauli Y matrix is usually represented as:

$$Y = \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \tag{2.6}$$

The Pauli Y Logic Gate is as shown in Figure 2.4

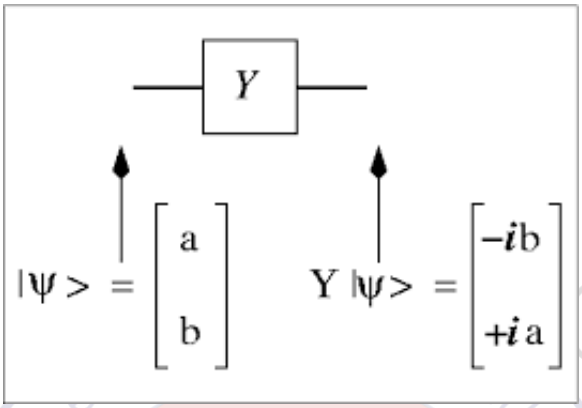


Figure 2.4: Pauli Y Logic Gate

3. **Pauli Z gate** - This gate only affects one qubit. It translates $|+\rangle$ to $|-\rangle$ and $|-\rangle$ to $|+\rangle$.Pauli X matrix is usually represented as:

$$Z = \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.7}$$

The Pauli Z Logic Gate is as shown in Figure 2.5

Figure 2.5: Pauli Z Logic Gate

4. **Hadamard Gate** - This gate works with a single qubit. It maps the basic state $|0\rangle$ to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|1\rangle$ to $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$, implying that the measurement will have an equal chance of yielding 1 or 0, i.e. it forms a superposition. The Hadamard matrix is used to depict it. It is represented by the Hadamard matrix:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{2.8}$$

The Hadamrad Logic Gate is as shown in Figure 2.6

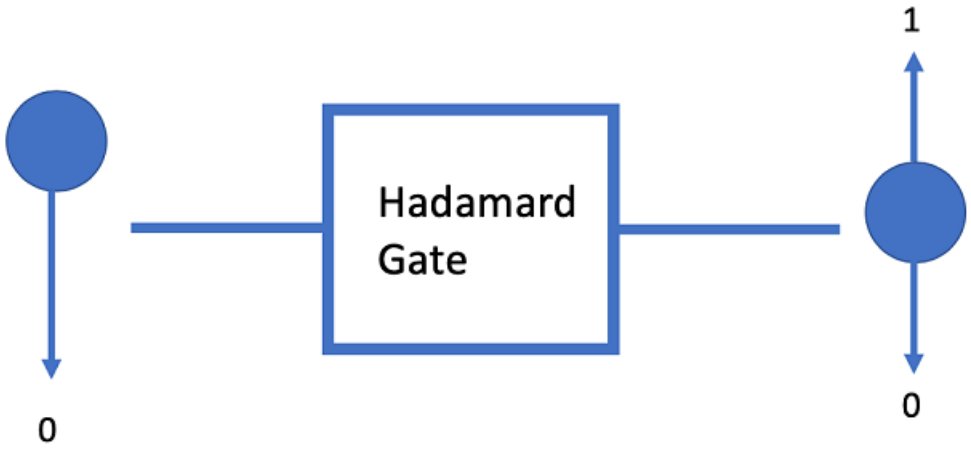


Figure 2.6: Hadamard Logic Gate

5. **Controlled NOT (CNOT) gate** - It has two qubits and acts on them. It's a quantum version of the XOR gate.With respect to the basis $|00\rangle, |01\rangle, |10\rangle, |11\rangle$,

   It is represented by the matrix:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.9}$$

   The CNOT gate can be described as the gate that maps the basis states $|a, b\rangle \mapsto |a, a \oplus b\rangle$, where $\oplus$ is XOR. The truth table for CNOT gate is shown in Table 2.1.

Table 2.1: Truth Table for CNOT gate

Input		Output	
X	Y	X	$X \oplus Y$
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

The CNOT Gate is as shown in Figure 2.7

2.3 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 system has abilities to simulate Quantum Mechanical

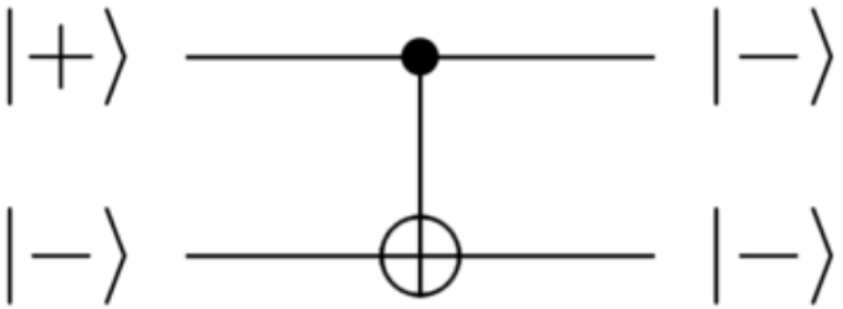


Figure 2.7: Controlled NOT Logic Gate

systems[19]. As compared with classical computers,A quantum computer based on quantum mechanics laws could more precisely and productively simulate these systems.

In the generally brief time frame from that point forward good amount of improvement has been done towards this objective. Issac Chuang et al. first spearheaded the utilization of Nuclear Magnetic Response (NMR) to build a Quantum computer which exhibited Shor's Factoring Algorithm [20]. This NMR-based quantum computer factored the number 15 into its prime factor components using 7 qubits. i.e. 3 and 5. NMR Spectroscopy is a chemical process for determining the structures of molecules by quantifying their properties in fluids, solids, and gases. This system utilizes nuclear spin of system of particles by applying a radio frequency electromagnetic pulse. The goal of this procedure is to control and identify the nuclear spins of the particles to make a reverberation recurrence that can be estimated. These spins of particles forms the bases of qubits which essentially becomes the quantum computer.

The requirement for better quantum algorithms have been increase in the past decade as there has been good advancement in physical implementation of quantum bits. Several algorithms have already been developed which are better than its classical counterparts. Deutsch's Algorithm and the Deutsch-Jozsa Algorithm is the first and the most simple quantum algorithm which first demonstrated quantum parallelism. Furthermore, Grover's Algorithm has been developed to search unsorted database and Grover's Algorithm designed for factoring number into its prime components.

2.3.1 Quantum Parallelism

Quantum Parallelism is an ability of Quantum Computers to exploit the properties of superposition and hence provide an advantage over classical counterparts. Any classical computer accesses only one value of a function $f(x)$ at one time. Conversely, Quantum computers evalu-

ate multiple values of the function $f(x)$ at the same time.

2.3.2 Deutsch-Jozsa Algorithm

David Deutsch and Richard Jozsa proposed the first deterministic quantum algorithm known as Deutsch-Jozsa (D-J) Algorithm in 1992 with improvements by other researchers like Richard Cleve, Artur Ekert in 1998 [21]. This was the first quantum algorithm which performed better than deterministic classical algorithms. The motivation behind this algorithm was to solve a black box problem which quantum computers can solve very efficiently when compared with classical computers which will take large number of queries to black box to solve that particular problem.

Given a hidden Boolean function f, which takes string of bits as inputs and present either 0 or 1 as its output i.e.

$$f(\{x_0, x_1, x_2, ...\}) \rightarrow 0 or 1 \tag{2.10}$$

where x_n is 0 or 1. This algorithm solves $f(x)$ in a single iteration and checks whether the given function $f(x)$ is constant or balanced. The function will be considered as constant if it returns all 0's or all 1's for any input. In contrast, the function will be considered as balanced if it returns 0's for exactly half the inputs and 1's for the other half.

While solving this problem on classical computer two scenarios will be considered. First, in the best case scenario, two queries to the oracle can determine it is balanced as if the output received is $f(0,0,0,...) \longrightarrow 0$ and $f(1,0,0,..) \longrightarrow 1$ since two different outputs have been obtained. In the worst case scenario, exactly half of all possible inputs plus one should be checked to determine whether the function is constant or not. For n qubits, number of possible inputs will be 2^n which implies at least $2^{n-1} + 1$ trials is required to determine if it is constant.

The same problem can be solved on a quantum computer with 100% confidence with only one call to the function $f(x)$. The given function is implemented as a quantum oracle to map state $|\ x\rangle\ |\ y\rangle$ to $|\ x\rangle\ |\ y \oplus f(x)\rangle$ where $\oplus$ is the addition modulo. Figure 2.8 shows the generic quantum circuit of the D-J algorithm.

The steps of the algorithm is described as follows:

1. Initially two quantum registers are prepared. The first register is an n-qubit register with initial value as $|\ 0\rangle$ and the second single qubit register is initialized to $|\ 1\rangle$.

$$|\ \psi_0\rangle = |\ 0\rangle^{\oplus n}\ |\ 1\rangle \tag{2.11}$$

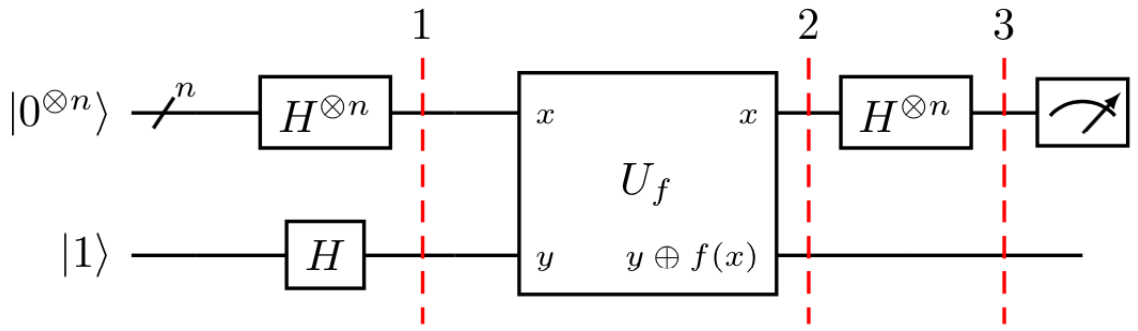


Figure 2.8: Generic circuit of Deutsch Jozsa Algorithm [22]

2. Hadamard gate is applied to each register. N bit Hadamard gate is applied to the first and single bit gate is applied to the second. This gate maps $|0\rangle$ to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|1\rangle$ to $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$.

$$|\psi_1\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} |x\rangle (|0\rangle - |1\rangle) \tag{2.12}$$

3. The quantum oracle which maps $|x\rangle|y \oplus f(x)\rangle$ is then applied to the outputs of the Hadamard gate.

$$\begin{aligned} |\psi_2\rangle &= \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} |x\rangle(|f(x)\rangle - |1 \oplus f(x)\rangle) \\ &= \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle(|0\rangle - |1\rangle) \end{aligned} \tag{2.13}$$

as for each x, $f(x)$ is either 0 or 1.

4. The single qubit register is ignored and n bit hadamard get is applied to the output of oracle. The results obatained is as follows:

$$\begin{aligned} |\psi_3\rangle &= \frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{f(x)} \left[\sum_{y=0}^{2^n-1} (-1)^{x.y} |y\rangle \right] \\ &= \frac{1}{2^n} \sum_{y=0}^{2^n-1} \left[\sum_{x=0}^{2^n-1} (-1)^{f(x)} (-1)^{x.y} \right] |y\rangle \end{aligned} \tag{2.14}$$

where $x.y = x_0y_0 \oplus x_1y_1 \oplus ....$ is the sum of bitwise product.

5. The first register is finally measured and if the probability evaluates to 1, the function $f(x)$ is constant else $f(x)$ is balanced.

2.3.3 Grover's Algorithm

Grover's Algorithm which is also referred as quantum search algorithm was first proposed by Lov Grover in 1996 [23]. This is a quantum algorithm used for unstructured search to find unique input to the black box function which produces a particular output value with high probability. This algorithm uses amplitude amplification trick to search for a particular element in unsorted database and hence provides quadratic speedup when compared to its classical counterparts. A. Ambainis et.al. particularly stated that algorithms for NP complete problems contains subroutine which acts as exhaustive search and hence Grover's algorithm can help in speeding up the search [24]. Grover's Algorithm is currently the best algorithm for 3SAT i.e. satisfiability problem.

The problem is defined as to find out whether a particular element or elements are present in a database which is a large list of unstructured elements. The input to Grover's algorithm is function $f : \{1, 2, ..., N\} \longrightarrow \{0, 1\}$. In the unstructured database analogy, the indices to a database is represented by domain and $f(x)$ results to 1 if and only if data points satisfies the criterion of search. The function f is accessed using the subroutine also called as oracle in form of unitary operator U_ω described as follows:

$$U_\omega|x\rangle = \begin{cases} |x\rangle & \text{if } x \neq \omega \\ -|x\rangle & \text{if } x = \omega \end{cases} \tag{2.15}$$

Solving this problem in classical computer will be highly computation extensive. There are numerous search algorithms that can speedup the search in classical computers but still will take more time when compared to quantum computers. The marked item to be searched in a unsorted database will require on an average N/2 queries to the function and at most will be have to be searched for all N elements in the worst case scenario.

Contrary to this, using Grover's Amplification trick, the marked item can be found in around $\sqrt{N}$ steps. Hence providing a quadratic speedup which is indeed a substantial time-saver for finding marked items in long lists. Since, there is no idea about the location of marked item hence guess of its location is expressed in terms of uniform superposition. In order to enhance the probability of the marked element a procedure called as Amplitude Amplification is used. This procedure amplifies the amplitude of the marked item which shrinks the other items amplitude in order to return the right item with high probability when measuring the final state. Amplitude Amplification steps are described as follows:

1. This procedure starts out in the uniform superposition state with equal probability to all the elements constructed from $|s\rangle = H^{\oplus n}|0\rangle^n$ as shown in Figure 2.9.

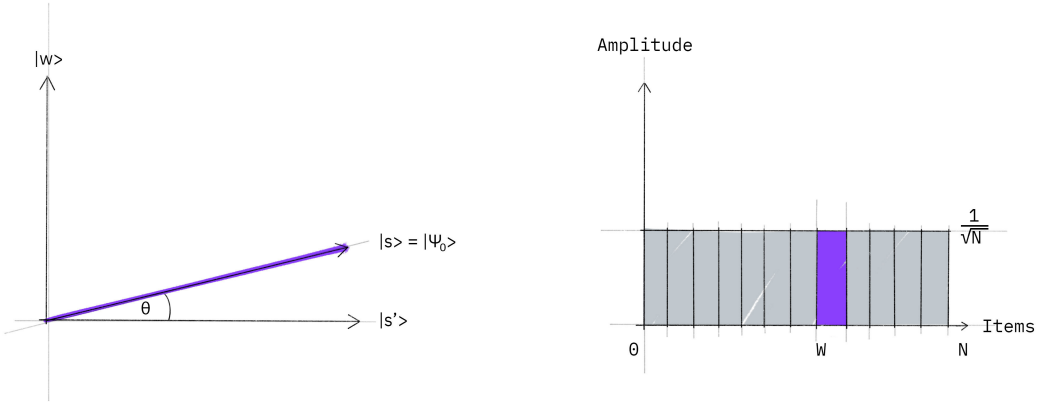


Figure 2.9: Step 1 of Amplitude Amplification

2. The reflection oracle U_f is applied to the state $|s\rangle$. This step negates the amplitude of the marked item which in turn reduces the average amplitude as shown in Figure 2.10.

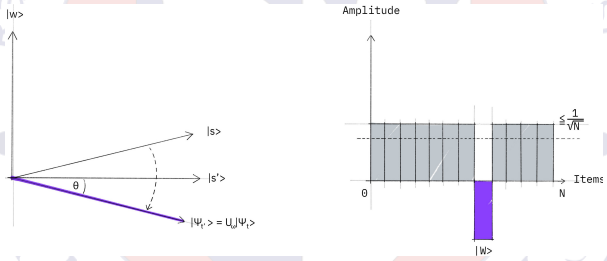


Figure 2.10: Step 2 of Amplitude Amplification

3. Now additional reflection is applied which rotates the element and the initial state reaches closer towards the winner. Step 2 and 3 is repeated for roughly $\sqrt{N}$ times to complete the transformation as denoted in Figure 2.11.

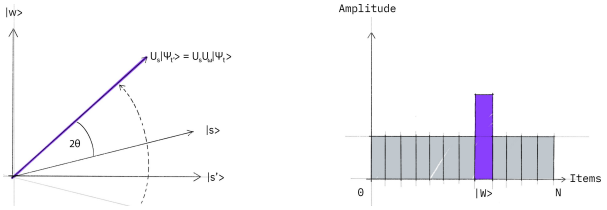


Figure 2.11: Step 3 of Amplitude Amplification

Figure 2.12 shows the Quantum circuit for Grover's Algorithm.

The steps of the algorithm is described as follows:

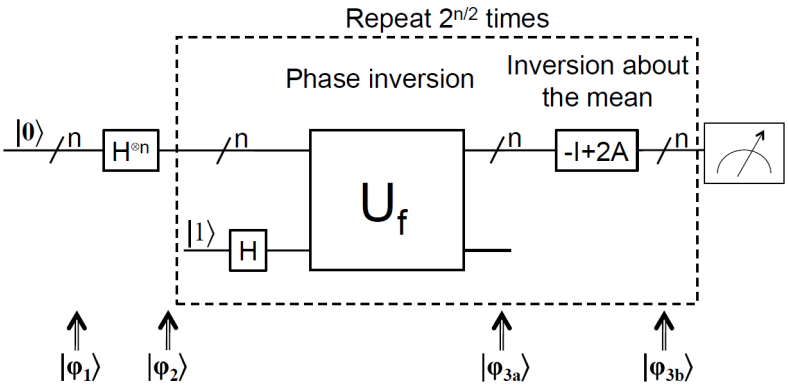


Figure 2.12: Generic circuit of Grover Algorithm [25]

1. Initially one quantum register is prepared to store n qubits all initialised to $|0\rangle$.

$$|\psi_0\rangle = |0\rangle^{\oplus n} \tag{2.16}$$

2. N bit Hadamard gate is applied to the register which maps $|0\rangle$ to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|1\rangle$ to $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$.

$$|\psi_1\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} |x\rangle (|0\rangle - |1\rangle) \tag{2.17}$$

3. The Quantum oracle is then applied to the outputs of the Hadamard gate with $f(x) = 1$ if x is the winning element and $f(x) = 0$ if x is otherwise.

4. The output is then applied to the diffuser which has two hadamard gate with a phase oracle in between with $f_o(x) = 0$ if $x = \{0, 0, ..., 0\}$ and $f_o(x) = 1$ if x is otherwise.

5. The phase oracle and diffuser is repeated 'r' times with

$$r = \frac{\pi}{4arcsin\frac{1}{\sqrt{2^N}}} - \frac{1}{2} \tag{2.18}$$

6. The output of this is finally measured with the winning element having the highest probability.

2.3.4 Shor's Algorithm

This algorithm was proposed by Peter Shor which was used to factor number into its prime components [26]. It is a polynomial time quantum computer algorithm for integer factorization

[27]. The most popular classical factoring algorithm requires super polynomial time to factor the product of two primes, the widely used cryptographic system Rivest-Shamir-Adleman (RSA) depends on considering being unthinkable to factor large numbers. Shor's Algorithm uses combination of Quantum Fourier Transform (QFT) and Quantum Phase Estimation (QPE) to find the factor of an integer on quantum computer. The efficiency of the QFT and modular exponentiation by repeated squaring helps in increasing the efficiency of Shor's Algorithm. Shor's Algorithm was first demonstrated by group at IBM. The group factored the number 15 using a quantum computer implemented using NMR with 7 qubits. Figure 2.13 denotes a basic flowchart showing an overview of Shor's Algorithm.

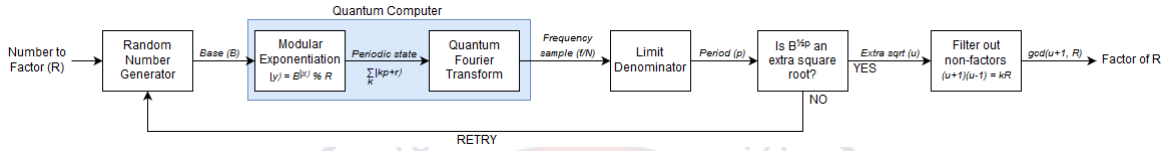


Figure 2.13: Flow Chart of Shor's Algorithm

The problem being solved using this algorithm is, given a composite number N, to find a non-trivial divisor of N which should be strictly between 1 and N. At first, a quick primality-testing algorithm is applied to verify whether number is prime or composite. N should be odd and need not be any power of a prime number. The factoring problem is transformed into period finding problem in polynomial time and efficient period finding algorithm is applied.

Solving this on classical computer requires highly extensive computations and will require super polynomial time to solve the problem. The quantum computer solves this in two steps:

1. Reduction of the factoring problem to the problem on order finding can be done on classical computer.

2. This order finding problem will then be solved using quantum algorithm.

Quantum computer approach will allow prime factorization of big integers to be performed in polynomial time. , on the order of $O(logn)^3$ where n is number of bits to be factored which is exponentially faster than best known classical algorithms which can factor in time $e^{\theta(n^{\frac{1}{3}} log^{\frac{2}{3}} n)}$. Quantum circuit for Shor's Algorithm is as shown in Figure 2.14.

2.4 Challenges and Future of Quantum Computers

There are various challenges to solve before a workable large-scale quantum computer may be realised. The fact that Quantum states have a short lifespan. and sensitive to contamination

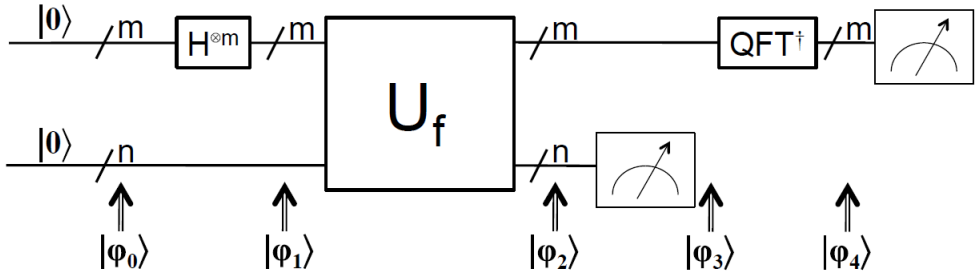
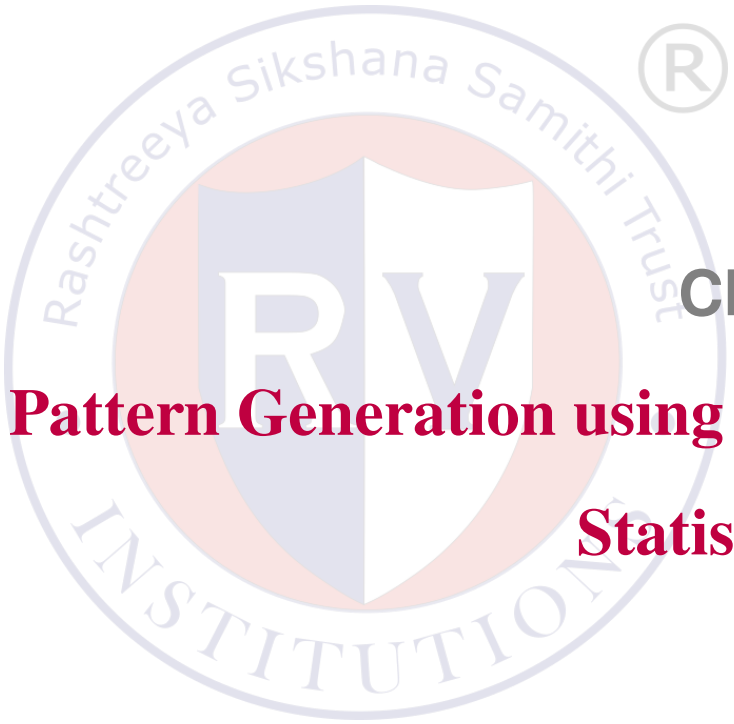


Figure 2.14: Quantum Circuit of Shor's Algorithm

from environmental noise is a big challenge. Quantum error correction was developed as a solution to this problem. Quantum mistake correction, on the other hand, comes at a significant price.Because of the expense of quantum error correction, instead of requiring on the order of 2048 qubits to factor a 2048-bit integer, ion trap technology is expected to require 1M qubits and 10M quantum gates.

It's difficult to say whether or not bigger-scale, fault-tolerant quantum computers will be developed, or when they will be produced. Moore's Law was projected to stop a long time ago, but inventive people managed to keep it going.Moore's Law was continually stretched by developers. Before a decision can be made, a lot of work must be done.It is possible to fully develop large-scale, fault-tolerant quantum computing. This isn't to say it can't happen. Modest-scale quantum computes, such as Shor's Algorithm, have been created to implement quantum algorithms with small input sizes. Quantum computer implementations are anticipated to rely on the same miniaturisation technology and manufacturing infrastructure as classical computer chip implementations on silicon wafers (i.e., quantum computers might be manufactured on wafers using the same manufacturing procedures as conventional computer chips). Some design ideas that apply to classical computer chips may also apply to quantum computer processors.

In this chapter a detailed introduction to Quantum computer has been discussed with explanation on transforming bits to qubits. This chapter also discusses Quantum Gate model with different logic gates and its comparison with classical counterparts. Various quantum algorithms have been discussed and its implementation on Quantum computers.



Chapter 3

Test Pattern Generation using Boolean Statisfiability

CHAPTER 3

TEST PATTERN GENERATION USING BOOLEAN STATISFIABILITY

Automatic Test Pattern Generator (ATPG) systems distinguishes defective components from defect-free components by generating set of inputs that cause outputs of component under test to be different if the component is defective. This chapter discusses the importance of ATPG systems in modern day VLSI circuits and problem related to generating test patterns. The chapter also elaborates on defining Boolean Satisfiability Test (SAT) and generating test patterns for single stuck at faults using SAT algorithms. This chapter concludes with comparison of different solutions to this problem.

3.1 Importance of ATPG systems

ATPG is an electrical design automation method/technology for determining an input sequence that allows automatic test equipment to discriminate between correct circuit behaviour and defective circuit behaviour caused by flaws when applied to a digital circuit. The patterns formed are used to test semiconductor devices after they have been manufactured or to help determine the reason of failure. The number of detected modelled faults, or fault models, and the number of generated patterns are used to determine ATPG's effectiveness. These metrics are used to determine the quality of a test and the length of time it takes to complete it (higher with more patterns). Another key factor to consider is ATPG efficiency. Automatic Test Pattern Generation (ATPG) is a critical issue in VLSI testing since it allows a designer to construct tests automatically to check for manufacturing flaws in a design. The end result is a tool that is roughly 30 % faster than a commercial ATPG solution. Automatic test pattern generation (ATPG) 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 structural methods, which do a topological search of the circuit under test, and the algebraic techniques, which construct test patterns by manipulating algebraic formulas, are the two types of algorithmic ATPG systems currently available for single stuck-at errors in combinational circuits.

A test pattern for a potentially defective circuit is a set of inputs that will cause the circuit outputs to differ if the circuit is defective vs if it is not. To derive the input set, a model of the circuit's probable defects (faults) is required. Then, using the single stuck-at

approach, which is the most popular among existing testing methods. With the exception of one wire attached to either a logic 0 or a logic 1, a defective circuit is expected to behave as if it were defect-free under this model (instead of correctly varying as a function of the circuit inputs).Inputs that are logically equal may fail in different ways. To create a test pattern for a circuit with a wire stuck at 1,it is must to make sure that the wire in issue would take on the logic value 0 in a properly working circuit. If this isn't the case, the circuit outputs will be the same whether the circuit is working or not, because the faulty and working circuits will have the same values. A disparity occurs when a line has a different value in the faulty and unfaulted circuits. The task of creating test patterns is known to be NP-complete. This does not imply that the problem should be abandoned. Various approaches is then required for test pattern generation to exhibit outstanding expected case behaviour on real circuits.

3.2 Boolean Satisfiability Problem

3.2.1 Introduction

SAT also referred to as propositional satisfiability issue in software engineering terms is the problem of whether there exists a translation that fulfills a given Boolean expression. All in all, it finds out if the variables of the given boolean equation can be reliably supplanted by the values TRUE or FALSE so that the formula evaluates to TRUE. Assuming this is the case, the condition is called satisfiable. Conversely, if no such interpretation exists, the limit imparted by the equation is FALSE for all conceivable variable errands and the expression is unsatisfiable.

SAT is the primary issue that was demonstrated to be NP-finished problem as per Cook-Levin speculation [28]. This suggests that all issues in the unpredictability class NP, which joins a wide extent of typical decision and headway issues, are all things considered as difficult to address as SAT. There is no known calculation that effectively take care of each SAT issue, and it is generally acknowledges that no such estimation exists and yet this conviction has not been demonstrated numerically which makes it a praised open issue in the speculation of mathematical calculations.

Heuristic SAT algorithms can take care of issue occurrence including a huge number of factors and recipes comprising million of symbols,[29] which is adequate for some down to eath SAT issues from, e.g., man-made brainpower, circuit design,[30] and customized speculation illustrating.

3.2.2 Generation of test pattern using Boolean Satisfiability Method

Boolean Satisfiability Problem has proved to be useful in various applications ranging from testing for VLSI circuits to machine learning. It's unique ability of solving problems based on pre-defined constraints solved a wide variety of possible solutions to combinatorial problems like scheduling courses in university and many more. Due to rapid increase in demand for semiconductors, highly efficient and defect free chips are the need of hour as these chips are the core of many crucial problems. Dependency on ATPG systems have increased in the past decade in order design defect free components which burdens these with their effectiveness in generation of test patterns. These systems have been running on various algorithms which have proved to be efficient but still does not meets its demands. Hence, generating boolean expression to test faults using SAT has proved to be useful.

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 single stuck at fault model.

2. Further after generating the expression, 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, carry circuit of the full adder is considered. Figure 3.1 shows the carry part of the full adder circuit. The boolean expression for the carry circuit is

$$Carry = A * B + C(A \oplus B) \tag{3.1}$$

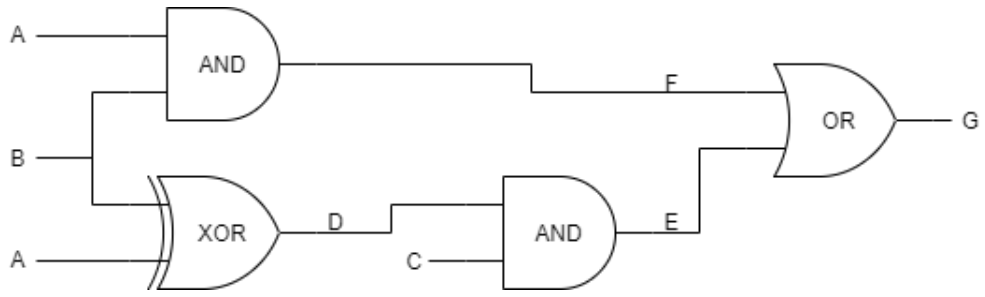


Figure 3.1: Circuit Diagram of Carry

In order to have a topological depiction of the circuit, Directed Acyclic Graph (DAG) are 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 3.2.

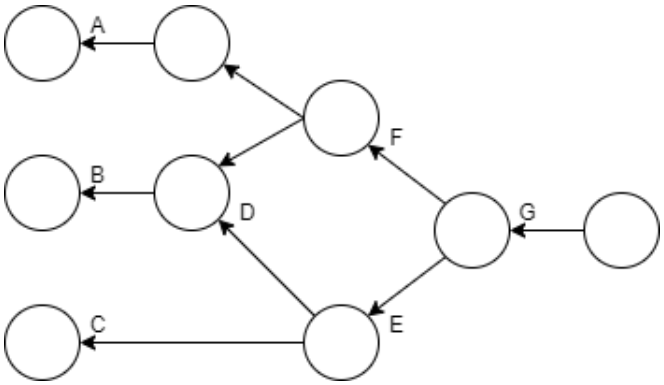


Figure 3.2: Directed Acyclic Graph of Carry

The expression will be drafted in three-component Conjunctive Normal Form (CNF) since it is very easy to control them algorithmically. CNF is also known as Product of Sums form in literal terms. Every term in this literal has atmost three components. For example, in order to get the three component CNF of OR gate following procedures are carried out:

1. An identity $A = B \equiv (A \implies B).(B \implies A)$ is applied to the boolean expression of OR gate i.e. $Z = P + Q$ to transform it to:

$$(Z \implies (P + Q)) \cdot ((P + Q) \implies Z) \tag{3.2}$$

2. Next the identity $A \implies B \equiv \bar{A} + B$ is applied and the transformation can be seen as:

$$Z \implies (P + Q) \equiv (\bar{Z} + P + Q) \tag{3.3}$$

$$(P + Q) \implies Z \equiv (Z + \bar{P}) \cdot (Z + \bar{Q}) \tag{3.4}$$

3. Hence after applying both transformation, the final three - component CNF of OR Gate is as follows:

$$(Z + \bar{X}) \cdot (Z + \bar{Y}) \cdot (\bar{Z} + P + Q) \tag{3.5}$$

Similarly, CNF of other basic gate can be calculated and is as shown Figure 3.3.

Since, the CNF expression for all basic gates are known, the expression for the carry circuit can be designed. The characteristic formula of any circuit can be extracted using the DAG

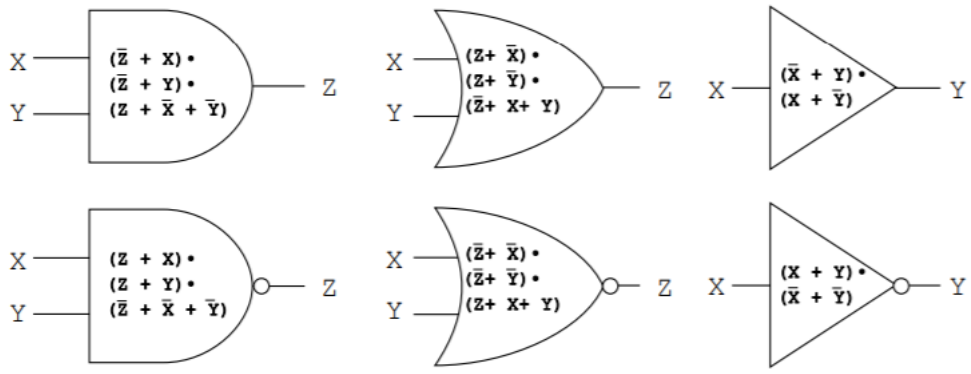


Figure 3.3: CNF for basic gates

since the fanout points and the gates labelled with an expression by just keeping output node as start point and strolling the graph. This is done by taking the combination of the entirety of the equations for all these visited nodes. Hence, the characteristic formula for the carry circuit is derived as follows:

$$(G + \bar{F}) \cdot (G + \bar{E}) \cdot (\bar{G} + F + E) \cdot (\bar{E} + C) \cdot (\bar{E} + D) \cdot (E + \bar{C} + \bar{D}) \cdot (D + \bar{A} + B) \cdot (D + A + \bar{B}) \cdot$$

$$(\bar{D} + A + B) \cdot (\bar{D} + \bar{A} + \bar{B}) \cdot (\bar{F} + A) \cdot (\bar{F} + B) \cdot (F + \bar{A} + \bar{B})$$

This characteristic formula is also the algebraic expression for the fault-free carry circuit. Figure 3.4 shows the fault-free circuit with characteristic formula of all its gates.

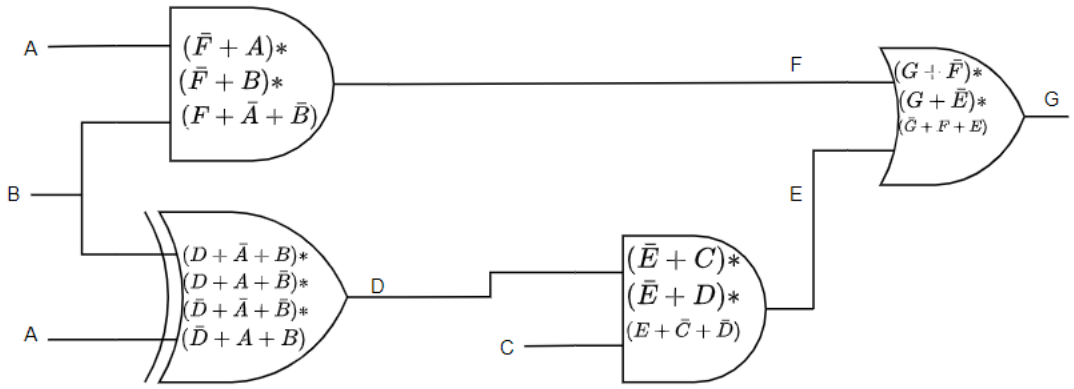


Figure 3.4: Carry Circuit with labelled gates

The faulty version of this circuit can be represented as by making a duplicate of this circuit with inserting two new node points where stuck at fault has effected keeping other nodes or variables intact. The faulty value will be generated at the site of the fault but other values will remain the same. In this example, node E has been considered as the fault site where the

value of E is always stuck at 1. Hence, the new expression for the faulty circuit will have to be generated. Since the E node was not behaving properly hence, E' has been applied at the input of the OR gate. Since the input E only effects the output of OR gate and hence the output has also been changed to G'. Both the faulty and fault-free carry circuit have indistinguishable conduct apart from those nodes where 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

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

The faulty circuit of the given carry circuit is represented in Figure 3.5.

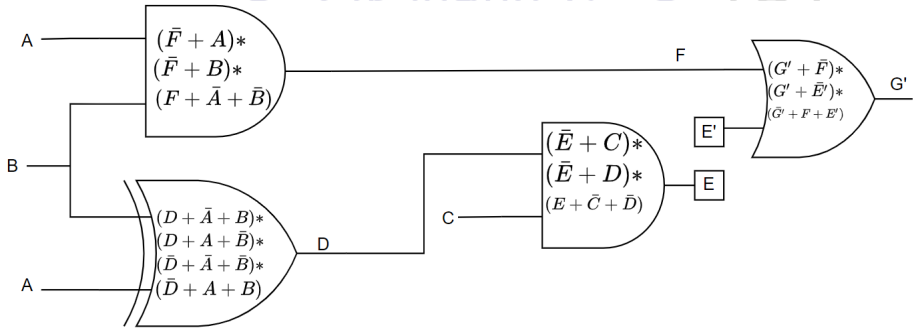


Figure 3.5: Carry Circuit with E stuck at 1

In order to test the given flaw, a set of inputs need to be found out for which yielded the faulty output to be different from the fault-free output. Adding an extra XOR gate between the two output can help to check for what set of inputs both faulty and fault-free output is different. In order to have the final formula, conjunction of the two expression is taken along with it added XOR formula is also added. The resulting CNF formula considering both faulty and fault-free outputs is as follows:

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

$$(G + \bar{F}) \cdot (G + \bar{E}) \cdot (\bar{G} + F + E) \cdot (\bar{E} + C) \cdot (\bar{E} + D) \cdot (E + \bar{C} + \bar{D}) \cdot (D + \bar{A} + B) \cdot (D + A + \bar{B}) \cdot$$

$$(\bar{D} + A + B) \cdot (\bar{D} + \bar{A} + \bar{B}) \cdot (\bar{G} + G' + out) \cdot (G + \bar{G'} + out) \cdot (\bar{G} + \bar{G'} + o\bar{u}t) \cdot (G + G' + o\bar{u}t)$$

The final circuit with faulty and fault free output of the resulting expression is as shown in Figure 3.6.

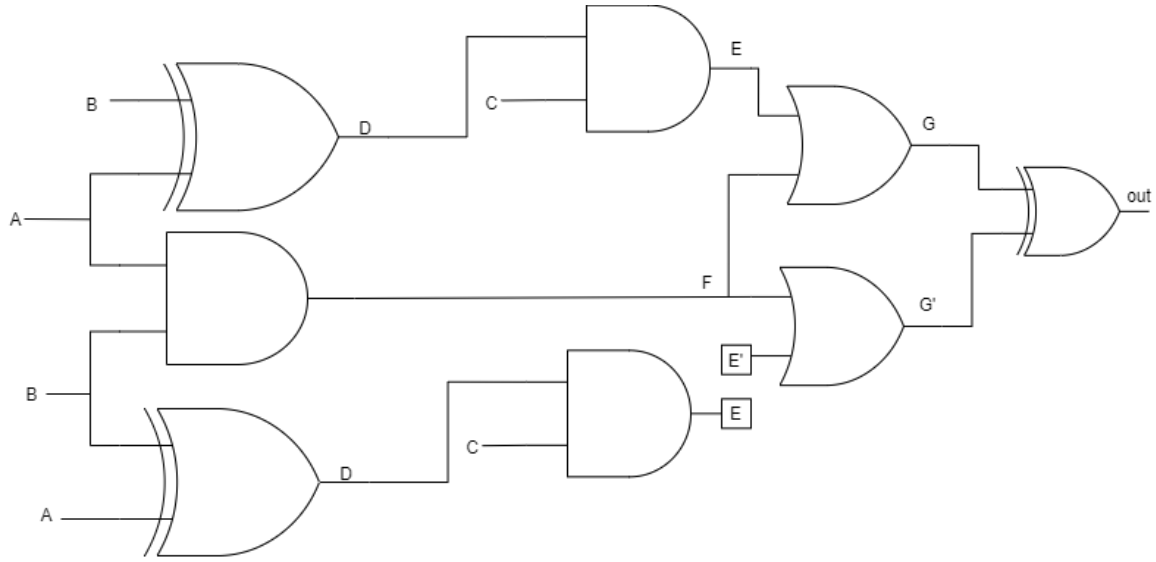


Figure 3.6: Final carry circuit with XOR gate whose output is 1‘

3.3 Solving Boolean Satisfiability Problem

As discussed in earlier sections regarding three component SAT, the issue of fulfilling a CNF function is NP complete issue[28]. Hence, using above method a transformation in issues has been achieved where initially in the worst case scenario the issue was being resolved in exponential time based on number of its inputs to another problem where this dependency is now based on circuit variables. Regardless, the class of expression generated using above method for combinational circuit is fascinating sub-class all three component CNF formulas. This reality can be used as an attempt to avoid all the worst case scenario[31]. Numerous scientists have given a perceived notion that typical direct of any three component SAT can be improvised significantly if the formula which needs to be solved is a part of restricted profile like formulas for combinational circuits [32].

3.3.1 Existing Algorithms

Once the extraction of the formula is completed, the next and the most important step is to solve the SAT equation using algorithm. At present, there are various algorithms which can solve SAT problem efficiently involving instances with thousands of variables. These SAT solvers are at present the most essential part of Electronic Design Automation (EDA) toolbox to automate and solve for various applications like VLSI testing, physical design, etc. Some of the existing ones are discussed below:

1. **DPLL SAT Solver** - The fundamental pursuit methodology was proposed in two papers in 1960s and is presently alluded to as Davis–Putnam–Logemann–Loveland (DPLL)

solver. Martin Davis et.al. in their paper [33] have utilized an orderly backtracking search technique to investigate the variable assignments in order for fulfilling assignments. Although this SAT solver is derived from DPLL algorithm, it is has improved efficiency for certain class of problems.

2. **Solving 3SAT using 2SAT** - As observed from the final logical expression that around 80 to 90% of the clauses have only two variables. Hence these 2CNF can be solved using 2SAT algorithms which have linear run time based on the number of clauses present in the final expression. 2SAT problem is solved using an algorithm proposed by B. Aspvall in the paper [34]. 2CNF is iterated throughout the circuit to solve 3CNF.Since 2CNF solver shows linear run and can be repeated many times to solve 3SAT also.

3.3.2 Solving Satisfiability Problem using Grover's Algorithm

As discussed in the previous chapter, Grover's search algorithm has proved to be the most efficient algorithm for searching for an element from an unsorted database. This algorithm uses amplitude amplification trick to find the target element. This algorithm can be utilized to look for answers for unstructured issues with a quadratic speed up as compared to its classical counterparts. An example of 3SAT problem in 3.6 is considered to explain the working of Grover's Algorithm on to solve SAT.

$$f(\alpha,\beta,\gamma) = (-\alpha \vee -\beta \vee -\gamma) \wedge (\alpha \vee \beta \vee -\gamma) \wedge (\alpha \vee \beta \vee \gamma) \tag{3.6}$$

In the above equation α, β, γ are the variables. Terms which are present inside the brackets are called clauses. The conjunction is represented by the '-' sign. This equation is satisfiable if for any value of variables $f(\alpha,\beta,\gamma) = 1$. These clauses are written in The Center for Discrete Mathematics and Theoretical Computer Science (DIMACS) CNF which is the format understood by the platform. The DIMACS CNF of the considered example is:

c example DIMACS CNF 3-SAT

p cnf 3 3

-1 -2 -3 0

1 2 -3 0

1 2 3 0

The first line is the comment. The next line indicates that the format is CNF along with number of variables and number of clauses or lines. Accordingly the next lines are written with a zero at

the end which indicates the completion of each line. This is then passed to the oracle function. 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.

This chapter discusses the importance of ATPG systems in present day chips and how test pattern can be generated using SAT. The boolean expression is evaluated for the carry circuit. Various SAT solvers have been discussed. Grover's Algorithm is the most efficient algorithm for unstructured database search and hence is used to find the result of the SAT problem.

Chapter 4

Implementation of Satisfiability Problem

CHAPTER 4

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[35]. The first section describes an introduction to the platform. The second section in the chapter discusses the flowchart of the algorithm along with its implementation.

4.1 IBM Quantum Experience

IBM Quantum Experience is the collective form of IBM Quantum Composer and IBM Quantum Lab. It creates an online stage permitting general public and premium admittance given by IBM for cloud-based quantum processing administrations and incorporates admittance to bunch of model quantum processors from IBM. Presently, IBM has added around 20 processors for the service available in different geographical locations with difference in number of qubits. Clients cooperate with a quantum processor through quantum circuits created programmatically using Jupyter Notebooks. These quantum circuits are compiled down to OpenQASM for execution on real processors [36].

4.2 Solving Boolean Satisfiability on IBM Q

As discussed in previous chapter, SAT expression evaluated from the circuit is solved using Grover's Algorithm which is implemented on IBM Quantum platform. 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). Figure 4.1 describes the flowchart of the algorithm. The steps of the algorithm are described as follows:

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. Code 4.1 indicates all the instances imported from various quantum libraries.

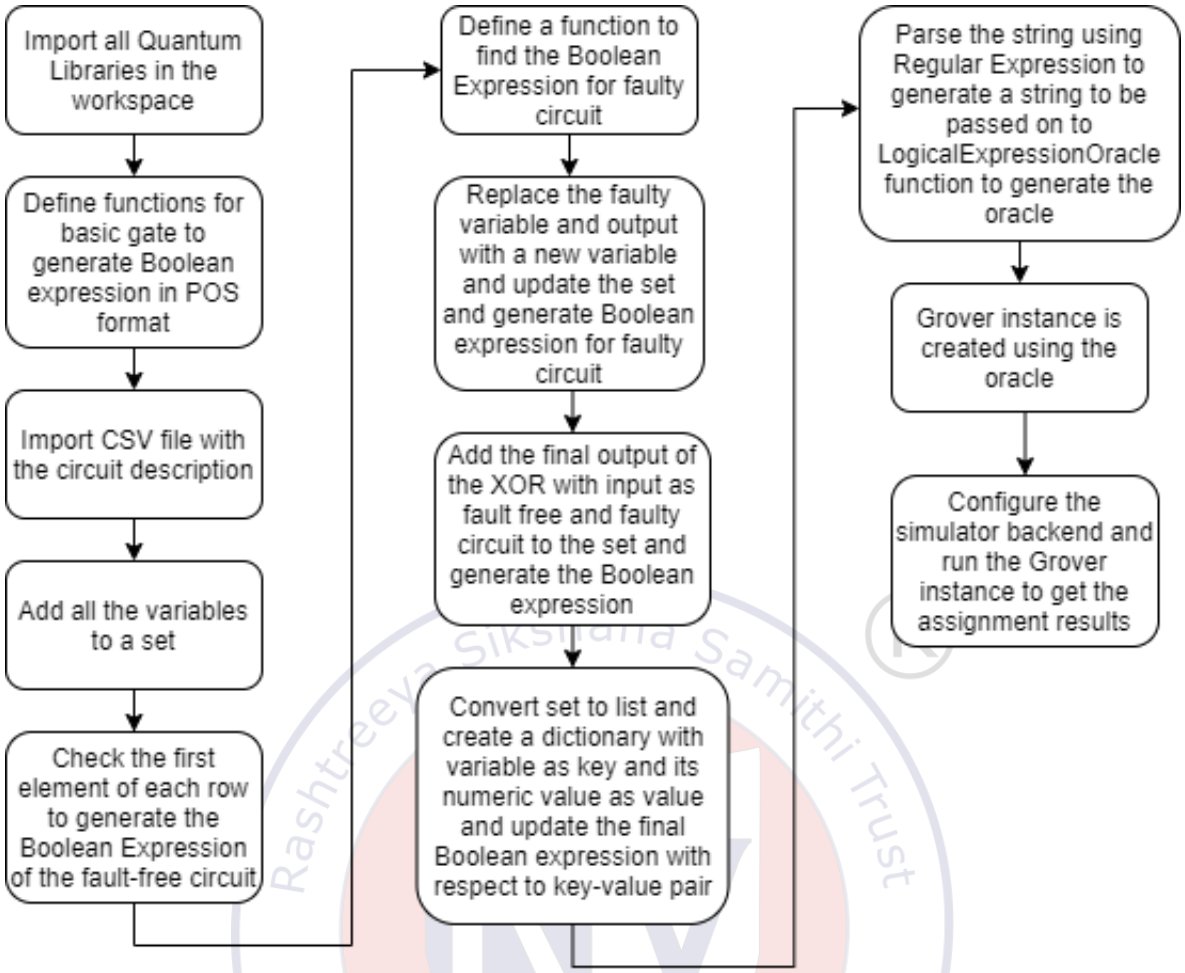


Figure 4.1: Flow Chart of the algorithm

```
import numpy as np
from qiskit import BasicAer
from qiskit.visualization import plot_histogram
from qiskit.aqua import QuantumInstance
from qiskit.aqua.algorithms import Grover
from qiskit.aqua.components.oracles import
   LogicalExpressionOracle, TruthTableOracle
```

Code 4.1: Importing libraries in workspace

2. **Expression for basic gates** - Once all the required libraries are imported, it is very important to define the logical expression of basic gate in the CNF form. Depending on type of gate in the considered circuit CNF expression will be written as per mentioned in Figure 3.3. Code 4.2 describes the code to generate logical expression for each gate in the CNF form. Function for each gate takes its inputs and output as arguments and

returns the logical expression.

```
def AND(a,b,c):
    return '({} + -{}) * ({} + -{}) * (-{} + -{} + {})'.format(a,c,b,c,a,b,c)
def OR(a,b,c):
    return "(-{} + {}) * (-{} + {}) * ({} + {} + -{})".format(a,c,b,c,a,b,c)
def NAND(a,b,c):
    return "({} + {}) * ({} + {}) * (-{} + -{} + -{})".format(a,c,b,c,a,b,c)
def NOR(a,b,c):
    return "(-{} + -{}) * (-{} + -{}) * ({} + {} + {})".format(a,c,b,c,a,b,c)
def NOT(a,b):
    return "({} + {}) * (-{} + -{})".format(a,b,a,b)
def XOR(a,b,c):
    return "(-{} + {} + {}) * ({} + -{} + {}) * (-{} + -{} + -{}) * ({} + {} + -{}))".format(a,b,c,a,b,c,a,b,c,a,b,c)
```

Code 4.2: CNF expressions for Basic gates

3. **Importing Circuit description file** The circuit description of the considered carry circuit in the Figure 3.1 has been specified in comma-separated values (CSV) format and is imported into the workspace to generate the expression. Table 4.1 demonstrates the CSV file of the carry circuit. This CSV file is imported and each row is read. All the variables

Table 4.1: 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

in the circuit are stored in a set which stores unique values. The process of importing the circuit description file and storing values in the set can be seen in Code 4.3.

```
import csv
my_set = set()
boolean_expression = " "
with open('Circuit Description.csv', 'r') as file:
    reader = csv.reader(file)
    for row in reader:
        if(row[0] == 'NOT'):
            my_set.add(row[1])
            my_set.add(row[3])
        else:
            my_set.add(row[1])
            my_set.add(row[2])
            my_set.add(row[3])
    ff_out = row[3]
my_set
```

Code 4.3: Importing circuit description file

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 expression for each row is appended to it to get the final boolean expression for the fault-free circuit. Code 4.4 describes the code to generate the final boolean expression for the given carry circuit in CNF format.

```
with open('Circuit Description.csv', 'r') as file:
    reader = csv.reader(file)
    items = []
    for row in reader:
        items
        if(row[0] == 'AND'):
            boolean_expression += AND(row[1],row[2],row[3])
        elif(row[0] == 'OR'):
            boolean_expression += OR(row[1],row[2],row[3])
        elif(row[0] == 'NAND'):
            boolean_expression += NAND(row[1],row[2],row[3])
```

```
        elif(row[0] == 'NOR'):
            boolean_expression += NOR(row[1],row[2],row[3])
        elif(row[0] == 'XOR'):
            boolean_expression += XOR(row[1],row[2],row[3])
        elif(row[0] == 'NOT'):
            boolean_expression += NOT(row[1],row[3])
        boolean_expression += ' * '
```

Code 4.4: Boolean expression for fault-free circuit

5. **Faulty circuit expression** - This steps 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 replaced 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. Code 4.5 shows the function for generating the boolean expression of faulty circuit which takes in two arguments the faulty variable and its value. It also updates the set with new variables. The function returns the boolean expression and the final output.

```
def faulty_ckt(variable,value):
    boolean_expression1 = "({}*)".format(variable)
    with open('Circuit Description.csv', 'r') as file:
        reader = csv.reader(file)
        for row in reader:
            if(row[1]==variable):
                boolean_expression1 += ' * '
                row[1] = "{}*".format(row[1])
                variable=row[3]
                row[3] = "{}*".format(row[3])
                if(row[0] == 'AND'):
                    boolean_expression1 += AND(row[1],row[2],row
                        [3])
```

```
13                  elif(row[0] == 'OR'):
14                      boolean_expression1 += OR(row[1],row[2],row
                            [3])
15                  elif(row[0] == 'NAND'):
16                      boolean_expression1 += NAND(row[1],row[2],
                            row[3])
17                  elif(row[0] == 'NOR'):
18                      boolean_expression1 += NOR(row[1],row[2],row
                            [3])
19                  elif(row[0] == 'XOR'):
20                      boolean_expression1 += XOR(row[1],row[2],row
                            [3])
21                  elif(row[0] == 'NOT'):
22                      boolean_expression1 += NOT(row[1],row[3])
23              elif(row[2]==variable):
24                  boolean_expression1 += ' * '
25                  row[2] = "{}*".format(row[2])
26                  variable=row[3]
27                  row[3] = "{}*".format(row[3])
28                  if(row[0] == 'AND'):
29                      boolean_expression1 += AND(row[1],row[2],row
                            [3])
30                  elif(row[0] == 'OR'):
31                      boolean_expression1 += OR(row[1],row[2],row
                            [3])
32                  elif(row[0] == 'NAND'):
33                      boolean_expression1 += NAND(row[1],row[2],
                            row[3])
34                  elif(row[0] == 'NOR'):
35                      boolean_expression1 += NOR(row[1],row[2],row
                            [3])
36                  elif(row[0] == 'XOR'):
37                      boolean_expression1 += XOR(row[1],row[2],row
                            [3])
```

```
                elif(row[0] == 'NOT'):
                    boolean_expression1 += NOT(row[1],row[3])
            if(row[0] == 'NOT'):
                my_set.add(row[1])
                my_set.add(row[3])
            else:
                my_set.add(row[1])
                my_set.add(row[2])
                my_set.add(row[3])
    return boolean_expression1,row[3]
```

Code 4.5: Boolean expression for faulty circuit

6. **Generating final expression** - This step is the most important where both the outputs of 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 is converted to 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. Code 4.6 demonstrates the generation of final boolean expression of the test circuit with E stuck at 1 fault.

```
faulty_boolean_exp,faulty_out = faulty_ckt('E',1)
my_set.add('out')
boolean_expression1 = " * "
boolean_expression1 += XOR(ff_out,faulty_out,'out')
faulty_boolean_exp += boolean_expression1
my_list = list(my_set)
my_list.sort()
my_list
a = {k: v for v, k in enumerate(my_list,start = 1)}
search_key = '*'
res = dict(filter(lambda item: search_key in item[0], a.items())
   )
```

```
keys_values = res.items()
new_d = {str(key): str(value) for key, value in keys_values}
for key in new_d.keys():
    faulty_boolean_exp = faulty_boolean_exp.replace(key, new_d[
        key])
boolean_expression += faulty_boolean_exp
keys_values = a.items()
new_d = {str(key): str(value) for key, value in keys_values}
new_d
for key in new_d.keys():
    boolean_expression = boolean_expression.replace(key, new_d[
        key])
```

Code 4.6: Final boolean expression of the test circuit

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 generate another string in the DIMACS CNF which will act as input to LogicalExpressionOracle to generate the oracle function for this particular SAT problem. The code mentioned in 4.7 executes this procedure to generate a final string.

```
import re
res = re.findall(r'\(.*?\)', boolean_expression)
new_str = ""
for i in range(len(res)):
    parse = re.findall(r'[-+]?[0-9]+', res[i])
    for j in range(len(parse)):
        new_str += '{} '.format(parse[j])
    new_str += "0\n"
input_3sat = '''
c example DIMACS-CNF 3-SAT
p cnf {} {}
'''.format(len(my_set),len(res))
input_3sat += new_str
```

Code 4.7: Generating string for oracle function

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 LogicalExpressionOracle. The oracle is passed to Grover function to create a Grover instance for the problem. The simulator background in the IBMQ 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. Code 4.8 describes the final code to run the Grover algorithm on SAT to get the required test pattern.

```
oracle = LogicalExpressionOracle(input_3sat)
grover = Grover(oracle)
from qiskit import IBMQ
IBMQ.save_account('API_token')
IBMQ.load_account()
provider = IBMQ.get_provider(hub='ibm-q',group='open')
sim = provider.get_backend('simulator_mps')
quantum_instance = QuantumInstance(sim, shots=8192)
result = grover.run(quantum_instance)
result['top_measurement']
qc = result['circuit']
qc.draw()
from qiskit.compiler import transpile
grover_compiled = transpile(result['circuit'],backend = sim,
   optimization_level =3)
print('gates = ',grover_compiled.count_ops())
print('depth = ',grover_compiled.depth())
```

Code 4.8: Executing Grover's Algorithm on SAT

This chapter describes each steps of the algorithm which was used to solve the Boolean Satisfiability Problem to generate test pattern to test the faulty circuit using Quantum algorithm. This chapter also indicates the output of this code which will be described in the next chapter.

Chapter 5

Results & Discussions

CHAPTER 5

RESULTS & DISCUSSIONS

The quantum implementation of Boolean Satisfiability problem for automatic test pattern generation systems is run on IBM quantum experience and results are verified in this chapter. The first section details the output of the algebraic expression for the circuit description and generates the CNF for each of the three cases considered. The second section shows the output running Grover's instance on the oracle function to search for the values which will satisfy the boolean expression. The final section compares the classical solution with the proposed quantum solution.

5.1 Output of the SAT problem

The most important step for finding the test pattern is to converting the test generation problem to boolean satisfiability problem and evaluating the boolean expression in the CNF format. The CSV file of the circuit description is read to generate the boolean expression in the Product of Sums form. Figure 5.1 shows the final boolean expression of the carry circuit in the POS form.

```
Out[36]: ' (-A + B + D) * (A + -B + D) * (-A + -B + -D) * (A + B + -D)) * (A + -F) * (B + -F) * (-A + -B + F) * (C + -E) * (D + -E) * (-
         C + -D + E) * (-E + G) * (-F + G) * (E + F + -G) * '
```

Figure 5.1: Evaluating boolean expression for carry circuit

Three different cases are considered based on the nets being near to the primary output of the circuit. The faulty boolean expression for all the three cases of stuck at fault are as follows:

1. **Net 'E' stuck at 1 fault** - Net 'E' is the net which is a step behind the primary input. The final boolean expression in the CNF format when variable 'E' is stuck at fault is as shown in the Figure 5.2. This final expression is then passed to the oracle function to create oracle of the Grover's Algorithm.

```
Out[40]: '\nc example DIMACS-CNF 3-SAT\np cnf 10 21\n-1 2 4 0\n1 -2 4 0\n-1 -2 -4 0\n1 2 -4 0\n1 -7 0\n2 -7 0\n-1 -2 7 0\n3 -5 0\n4 -5 0
         \n-3 -4 5 0\n-5 8 0\n-7 8 0\n5 7 -8 0\n6 0\n-6 9 0\n-7 9 0\n6 7 -9 0\n-8 9 10 0\n8 -9 10 0\n-8 -9 -10 0\n8 9 -10 0\n'
```

Figure 5.2: CNF expression for 'E' stuck at 1

It can be observed from the figure that the number of variables in this case is 10 and the number of literals present in the CNF equation is 21.

2. **Net 'D' stuck at 1 fault** - Net 'D' is considered as the second level as there exists two gates between Net 'D' and the primary output. Figure 5.3 shows the final boolean expression in the CNF format for net 'D' stuck at 1. The oracle function is called with this expression as an input to generate the Grover's oracle.

```
Out[9]: '\nc example DIMACS-CNF 3-SAT\np cnf 11 24\n-1 2 4 0\n1 -2 4 0\n-1 -2 -4 0\n1 2 -4 0\n1 -8 0\n2 -8 0\n-1 -2 8 0\n3 -6 0\n4 -6 0
        \n-3 -4 6 0\n-6 9 0\n-8 9 0\n6 8 -9 0\n5 0\n3 -7 0\n5 -7 0\n-3 -5 7 0\n-7 10 0\n-8 10 0\n7 8 -10 0\n-9 10 11 0\n9 -10 11 0\n-9
        -10 -11 0\n9 10 -11 0\n'
```

Figure 5.3: CNF expression for 'D' stuck at 1

It can be observed from the figure that the number of variables in this case is 11 and the number of literals present in the CNF equation is 24.

3. **Net 'A' stuck at 1 fault** - Net 'A' is the farthest net from the primary output and is considered as Level 3 as there is three gates between this net which is the primary input in this case and the primary output. Figure 5.4 shows the final boolean expression in the CNF format for the net 'A' stuck at 1. The oracle function takes this expression to generate the Grover's oracle.

```
Out[24]: '\nc example DIMACS-CNF 3-SAT\np cnf 12 28\n-1 3 5 0\n1 -3 5 0\n-1 -3 -5 0\n1 3 -5 0\n1 -9 0\n3 -9 0\n-1 -3 9 0\n4 -7 0\n5 -7 0
         \n-4 -5 7 0\n-7 10 0\n-9 10 0\n7 9 -10 0\n2 0\n-2 3 6 0\n2 -3 6 0\n-2 -3 -6 0\n2 3 -6 0\n4 -8 0\n6 -8 0\n-4 -6 8 0\n-8 11 0\n-9
         11 0\n8 9 -11 0\n-10 11 12 0\n10 -11 12 0\n-10 -11 -12 0\n10 11 -12 0\n'
```

Figure 5.4: CNF expression for 'A' stuck at 1

It can be observed from the figure that the number of variables in this case is 12 and the number of literals present in the CNF equation is 28.

5.2 Output of the Grover's Algorithm

The final boolean expression in the CNF format as generated in the previous step is passed to the oracle function which creates the Grover's oracle. This oracle is run using Grover's algorithm to find the test pattern which can satisfy the boolean satisfiability problem. The algorithm was run for all the three cases considered and the outputs are discussed as follows:

1. **Net 'E' stuck at 1 fault** - The CNF expression for this case is as shown in the previous step. This is then send to generate the Grover's oracle to run Grover's algorithm on it and generate the test pattern which can satisfy the given equation. Figure 5.5 shows the output after running the Grover's instance on this case. As seen in the above figure the first three bits are the input bits which are actually the test pattern to detect 'E' stuck at

```
/opt/conda/lib/python3.8/site-packages/qiskit/providers/ibmq/ibmqbackend.py:814: DeprecationWarning: Passing a Qobj to Backend.
run is deprecated and will be removed in a future release. Please pass in circuits or pulse schedules instead.
  return super().run(circuits, job_name=job_name, job_share_level=job_share_level,

[1, -2, 3, -4, -5, -6, 7, 8, 9, 10]
```

Figure 5.5: Test pattern for 'E' stuck at 1 fault

1 fault. The test pattern obtained is (1,2,3) i.e. 111 is the test pattern which can detect this fault. In the Grover's circuit for this case, the first 10 bits are the actual input bits and the remaining 41 bits are ancillia bits which are just the extra bits considered to perform quantum computations. This Grover's circuit can be further transpiled to know the number of reversible gate used to solve this problem. Figure 5.6 shows the type of gate and the count of each type of gate.

```
gates =  OrderedDict([('cx', 2010), ('u1', 1679), ('u2', 236), ('tdg', 123), ('t', 115), ('x', 49), ('u3', 14), ('measure', 1
0), ('h', 2)])
depth =  3312
```

Figure 5.6: Number of reversible gates for 'E' stuck at 1

2. **Net 'D' stuck at 1 fault** - The CNF expression for this case is as shown in the previous step. This is then send to generate the Grover's oracle to run Grover's algorithm on it and generate the test pattern which can satisfy the given equation. Figure 5.7 shows the output after running the Grover's instance on this case. As seen in the above figure the

```
/opt/conda/lib/python3.8/site-packages/qiskit/aqua/quantum_instance.py:135: DeprecationWarning: The class qiskit.aqua.QuantumIn
stance is deprecated. It was moved/refactored to qiskit.utils.QuantumInstance (pip install qiskit-terra). For more information
see <https://github.com/Qiskit/qiskit-aqua/blob/master/README.md#migration-guide>
  warn_class('aqua.QuantumInstance',
/opt/conda/lib/python3.8/site-packages/qiskit/providers/ibmq/ibmqbackend.py:814: DeprecationWarning: Passing a Qobj to Backend.
run is deprecated and will be removed in a future release. Please pass in circuits or pulse schedules instead.
  return super().run(circuits, job_name=job_name, job_share_level=job_share_level,

[-1, 2, -3, 4, 5, 6, 7, 8, 9, -10, 11]
```

Figure 5.7: Test pattern for 'D' stuck at 1 fault

first three bits are the input bits which are actually the test pattern to detect 'D' stuck at 1 fault. The test pattern obtained is (1,-2,3) i.e. 101 is the test pattern which can detect this fault. In the Grover's circuit for this case, the first 11 bits are the actual input bits and the remaining 47 bits are ancillia bits which are just the extra bits considered to perform quantum computations. This Grover's circuit can be further transpiled to know the number of reversible gate used to solve this problem. Figure 5.8 shows the type of gate and the count of each type of gate.

3. **Net 'A' stuck at 1 fault** - The CNF expression for this case is as shown in the previous step. This is then send to generate the Grover's oracle to run Grover's algorithm on it

```
gates =  OrderedDict([('cx', 3610), ('u1', 3241), ('u2', 270), ('tdg', 141), ('t', 117), ('x', 43), ('u3', 14), ('measure', 1
1), ('h', 3)])
depth =  5964
```

Figure 5.8: Number of reversible gates for 'D' stuck at 1

and generate the test pattern which can satisfy the given equation. Figure 5.9 shows the output after running the Grover's instance on this case. As seen in the above figure the

```
/opt/conda/lib/python3.8/site-packages/qiskit/providers/ibmq/ibmqbackend.py:814: DeprecationWarning: Passing a Qobj to Backend.
run is deprecated and will be removed in a future release. Please pass in circuits or pulse schedules instead.
  return super().run(circuits, job_name=job_name, job_share_level=job_share_level,

[1, 2, 3, -4, 5, 6, -7, -8, -9, -10, -11, 12]
```

Figure 5.9: Test pattern for 'A' stuck at 1 fault

first three bits are the input bits which are actually the test pattern to detect 'A' stuck at 1 fault. The test pattern obtained is (1,-2,-3) i.e. 100 is the test pattern which can detect this fault. In the Grover's circuit for this case, the first 12 bits are the actual input bits and the remaining 55 bits are ancillia bits which are just the extra bits considered to perform quantum computations. This Grover's circuit can be further transpiled to know the number of reversible gate used to solve this problem. Figure 5.10 shows the type of gate and the count of each type of gate.

```
gates =  OrderedDict([('cx', 6788), ('u1', 6348), ('u2', 324), ('tdg', 165), ('t', 133), ('x', 47), ('u3', 24), ('measure', 1
2), ('h', 4)])
depth =  11291
```

Figure 5.10: Number of reversible gates for 'A' stuck at 1

Table 5.1 summarises the results for all the three cases considered along with simulation time for each cases.

In the table, Level refers to the number of stages or gates between the considered net and the primary output. Based on three cases considered in this project, it can be observed that as the faulty net is very near to the primary input, less number of literals are present in the boolean expression which speeds up the simulation process. As the net starts getting far from the primary output, the number of literals in the expression starts increasing which not only increases the simulation time of this algorithm but also requires huge amount of quantum resources. The single bit increase in the variable list increases the ancillia bits i.e. the extra bits by 6. The same affects the simulation time where the best case simulation time for the case when the faulty net is very near to the primary output is less than a minute but the case for the primary input which

Table 5.1: Summarised Results of the three cases

Parameter	**'E' @ 1**	**'D' @ 1**	**'A' @ 1**
Level from the primary output	Level 1	Level 2	Level 3
Number of variables	10	11	12
Number of literals	21	24	28
Test pattern generated	111	101	100
Number of ancillia bits	41	47	55
Total number of reversible gate	4238	7450	13845
Simulation time in min	1	18	46

will take the most time to simulate takes around 45 min and requires around 4 times the number of gates as required in the prior case.

Apart from being computationally extensive, this is one of the method for automatic test pattern generation which can even cover the untestable and undetectable faults. Solving of SAT problem using Grover's algorithm has drastically reduced the run time as the same problem when considered in any of the best known classical algorithm will require huge amount of time when the number of bits are more than 10. Hence for small circuits this algorithm proves to be better both in run time and for finding the test patterns which can be able to detect any of the faults.

The following chapter details the results of each of the three cases considered based on the presence of net with respect to the primary output. The test pattern required to detect each fault has been calculated and the number of reversible gates required to solve each is also shown. The last table summarises the results for the entire three cases mentioning the simulation time of each one of them.

Chapter 6

Conclusion and Future Scope

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

QC is an interesting new field at the crossing point of Computer Science, Physics and Mathematics. QC is understood by learning the basic quantum logic circuits and various quantum algorithms. Later these circuits and algorithms were used to design a given objective function and the test pattern for single stuck at fault is determined by SAT algorithm. These Quantum Circuits were implemented on IBM Quantum Experience and the results were compared with respect to classical computation.

The whole problem of generating the boolean expression for single stuck at fault model in combinational circuit has been divided into two steps first the Boolean expression is emerged which will be defining the set of test structures for a single stuck at fault model and 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. Grover's search algorithm has been shown to be the most efficient approach for searching an unsorted database for an element. 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. 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 results are analysed.

The primary objective was to have a better understanding of different Quantum algorithms and and its applications to solve complex problems. Algorithms like Shor's and Grover's are proved to be very useful in problems like factorising large numbers or finding an element from unsorted database in a very efficient way respectively. This unique property of Grover's algorithm made it very useful to solve SAT problems which also have a wide range of applications in solving complex problems like ATPG. These systems have always been computationally very extensive hence its implementation on quantum computer have drastically reduced the run time to the factors of $\sqrt{N}$. This implementation also generates test pattern for faults which are undetectable. The boolean satisfiability problem when run on classical computer with the best search algorithm will have a super polynomial run time. The three cases considered details more about the effect on stages in any circuit. An increase in the variable doubles the ancillia

bits and increases the run time by a marginal factor.

6.2 Future Scope

Quantum computing can be a game-changer in fields like cryptography and chemistry, material science, agriculture, electronic design automation tool and pharmaceuticals. Quantum computing can also help deep learning and machine learning researchers by providing a speed up in the training and testing process. This project can be extended to generate test patterns for multiple stuck at faults and also large number of gates can be considered for analysis. The analysis of this project can also be extended to circuits having more than one primary outputs. Boolean Satisfiability Problem is beneficial in VLSI circuit testing to solve issues based on predefined restrictions in a wide range of combinational problems, including university course scheduling and many others.

6.3 Learning Outcomes of the Project

- Understanding Quantum computation and how it is different from classical computation.
- Understanding quantum logic circuits and quantum algorithms.
- Understanding the test pattern for single stuck at fault which is determined by SAT algorithm.
- Understanding Grover's algorithm implementation for Boolean satisfiabilty problem.
- Understanding the implementation of quantum circuits on IBM Quantum Experience.

BIBLIOGRAPHY

[1] M. Ostaszewski, E. Grant, and M. Benedetti, "Structure optimization for parameterized quantum circuits," *Quantum*, vol. 5, p. 391, 2021.

[2] L. Gyongyosi, "Objective function estimation for solving optimization problems in gate-model quantum computers," *Scientific reports*, vol. 10, no. 1, pp. 1–21, 2020.

[3] ——, "Unsupervised quantum gate control for gate-model quantum computers," *Scientific reports*, vol. 10, no. 1, pp. 1–16, 2020.

[4] L. Gyongyosi and S. Imre, "Quantum circuit design for objective function maximization in gate-model quantum computers," *Quantum Information Processing*, vol. 18, no. 7, pp. 1–33, 2019.

[5] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A survey of optimization methods from a machine learning perspective," *IEEE transactions on cybernetics*, vol. 50, no. 8, pp. 3668–3681, 2019.

[6] J. S. Chen, D. A. Desai, S. P. Heyns, and F. Pietra, "Literature review of numerical simulation and optimisation of the shot peening process," *Advances in Mechanical Engineering*, vol. 11, no. 3, p. 1 687 814 018 818 277, 2019.

[7] M. Alam, A. Ash-Saki, and S. Ghosh, "Analysis of quantum approximate optimization algorithm under realistic noise in superconducting qubits," *arXiv preprint arXiv:1907.09631*, 2019.

[8] T. Humble, "Consumer applications of quantum computing: A promising approach for secure computation, trusted data storage, and efficient applications," *IEEE Consumer Electronics Magazine*, vol. 7, no. 6, pp. 8–14, 2018.

[9] T. Haener, M. Soeken, M. Roetteler, and K. M. Svore, "Quantum circuits for floating-point arithmetic," in *International Conference on Reversible Computation*, Springer, 2018, pp. 162–174.

[10] E. Farhi, J. Goldstone, S. Gutmann, and H. Neven, "Quantum algorithms for fixed qubit architectures," *arXiv preprint arXiv:1703.06199*, 2017.

[11] M. Mariantoni, H. Wang, T. Yamamoto, M. Neeley, R. C. Bialczak, Y. Chen, M. Lenander, E. Lucero, A. D. O'Connell, D. Sank, *et al.*, "Implementing the quantum von neumann architecture with superconducting circuits," *Science*, vol. 334, no. 6052, pp. 61–65, 2011.

[12] G. Florio and D. Picca, "Quantum implementation of elementary arithmetic operations," *arXiv preprint quant-ph/0403048*, 2004.

[13] A. U. Khalid, Z. Zilic, and K. Radecka, "Fpga emulation of quantum circuits," in *IEEE International Conference on Computer Design: VLSI in Computers and Processors, 2004. ICCD 2004. Proceedings.*, IEEE, 2004, pp. 310–315.

[14] X. Zhou, D. W. Leung, and I. L. Chuang, "Methodology for quantum logic gate construction," *Physical Review A*, vol. 62, no. 5, p. 052 316, 2000.

[15] Z. Zilic and K. Radecka, "The role of super-fast transforms in speeding up quantum computations," in *Proceedings 32nd IEEE International Symposium on Multiple-Valued Logic*, IEEE, 2002, pp. 129–135.

[16] R. P. Feynman, "Simulating physics with computers," *International journal of theoretical physics*, vol. 21, no. 6/7, pp. 467–488, 1982.

[17] Subhash Kak, *A quantum computing future is unlikely, due to random hardware errors*, 2019. [Online]. Available: https://images.theconversation.com/files/303588/original/file-20191125-74588-1s17qy1.png?ixlib=rb-1.1.0&q=45&auto=format&w=1000&fit=clip.

[18] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667.

[19] R. P. Feynman, J. G. Hey, and R. W. Allen, *Feynman Lectures on Computation*. USA: Addison-Wesley Longman Publishing Co., Inc., 1998, ISBN: 0201386283.

[20] L. Vandersypen, M. Steffen, G. Breyta, C. Yannoni, M. Sherwood, and I. Chuang, "Experimental realization of shor's quantum factoring algorithm using nuclear magnetic resonance. nature, 414:883," *Nature*, vol. 414, pp. 883–7, Dec. 2001. DOI: 10.1038/414883a.

[21] D. Deutsch and R. Jozsa, "Rapid solution of problems by quantum computation," GBR, Tech. Rep., 1992.

[22] 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.

[23] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '96, Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996, pp. 212–219, ISBN: 0897917855. DOI: 10.1145/237814.237866. [Online]. Available: https://doi.org/10.1145/237814.237866.

[24] A. Ambainis, "Quantum search algorithms," *SIGACT News*, vol. 35, no. 2, pp. 22–35, Jun. 2004, ISSN: 0163-5700. DOI: 10.1145/992287.992296. [Online]. Available: https://doi.org/10.1145/992287.992296.

[25] C. Figgatt, D. Maslov, K. Landsman, N. Linke, S. Debnath, and C. Monroe, "Complete 3-qubit grover search on a programmable quantum computer," *Nature Communications*, vol. 8, Dec. 2017. DOI: 10.1038/s41467-017-01904-7.

[26] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997, ISSN: 0097-5397. DOI: 10.1137/S0097539795293172. [Online]. Available: https://doi.org/10.1137/S0097539795293172.

[27] P. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700.

[28] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, ser. STOC '71, Shaker Heights, Ohio, USA: Association for Computing Machinery, 1971, pp. 151–158, ISBN: 9781450374644. DOI: 10.1145/800157.805047. [Online]. Available: https://doi.org/10.1145/800157.805047.

[29] O. Ohrimenko, P. J. Stuckey, and M. Codish, "Propagation = lazy clause generation," in *Principles and Practice of Constraint Programming – CP 2007*, C. Bessière, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 544–558, ISBN: 978-3-540-74970-7.

[30] T. Hong, Y. Li, S.-B. Park, D. Mui, D. Lin, Z. A. Kaleq, N. Hakim, H. Naeimi, D. S. Gardner, and S. Mitra, "Qed: Quick error detection tests for effective post-silicon validation," in *2010 IEEE International Test Conference*, 2010, pp. 1–10. DOI: 10.1109/TEST.2010.5699215.

[31] M. Davis and H. Putnam, "A computing procedure for quantification theory," *J. ACM*, vol. 7, no. 3, pp. 201–215, Jul. 1960, ISSN: 0004-5411. DOI: 10.1145/321033.321034. [Online]. Available: https://doi.org/10.1145/321033.321034.

[32] P. W. Purdom and C. A. Brown, "Evaluating search methods analytically," in *In Proceedings of the National Conference on Artificial Intelligence*, 1982, pp. 124–127.

[33] M. Davis, G. Logemann, and D. Loveland, "A machine program for theorem-proving," *Commun. ACM*, vol. 5, no. 7, pp. 394–397, Jul. 1962, ISSN: 0001-0782. DOI: 10.1145/368273.368557. [Online]. Available: https://doi.org/10.1145/368273.368557.

[34] B. Aspvall, M. Plass, and R. Tarjan, "A linear-time algorithm for testing the truth of certain quantified boolean formulas," *Inf. Process. Lett.*, vol. 8, pp. 121–123, 1979.

[35] *IBM Quantum*, https://quantum-computing.ibm.com/, June,2021.

[36] *Qiskit: An open-source framework for quantum computing*, 2021. DOI: 10.5281/zenodo.2562110.

1RV17EC189_Nishant Agrawal_Dr Ramavenkateswaran

ORIGINALITY REPORT

15%	8%	11%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	qiskit.org Internet Source	1%
2	en.wikipedia.org Internet Source	1%
3	docplayer.net Internet Source	1%
4	Parhami, Behrooz. "Computer Architecture", Oxford University Press Publication	1%
5	www.springerprofessional.de Internet Source	1%
6	Laszlo Gyongyosi. "Objective function estimation for solving optimization problems in gate-model quantum computers", Scientific Reports, 2020 Publication	1%
7	Xinlan Zhou, Debbie W. Leung, Isaac L. Chuang. "Methodology for quantum logic gate construction", Physical Review A, 2000 Publication	1%

8	Laszlo Gyongyosi. "Unsupervised Quantum Gate Control for Gate-Model Quantum Computers", Scientific Reports, 2020 Publication	1 %
9	community.qiskit.org Internet Source	1 %
10	drrajivdesaimd.com Internet Source	1 %
11	Shiliang Sun, Zehui Cao, Han Zhu, Jing Zhao. "A Survey of Optimization Methods From a Machine Learning Perspective", IEEE Transactions on Cybernetics, 2020 Publication	1 %
12	science.sciencemag.org Internet Source	<1 %
13	www.cs.ox.ac.uk Internet Source	<1 %
14	Z. Zilic, K. Radecka. "The role of super-fast transforms in speeding up quantum computations", Proceedings 32nd IEEE International Symposium on Multiple- Valued Logic, 2002 Publication	<1 %
15	Laszlo Gyongyosi, Sandor Imre. "Quantum circuit design for objective function maximization in gate-model quantum	<1 %

computers", Quantum Information Processing, 2019
Publication

16	Vladimir Silva. "Practical Quantum Computing for Developers", Springer Science and Business Media LLC, 2018 Publication	<1 %
17	T. Larrabee. "Test pattern generation using Boolean satisfiability", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1992 Publication	<1 %
18	www.andrew.cmu.edu Internet Source	<1 %
19	Chunfeng Liu, Bing Li, Bhargab B. Bhattacharya, Krishnendu Chakrabarty, Tsung-Yi Ho, Ulf Schlichtmann. "Test Generation for Flow-Based Microfluidic Biochips with General Architectures", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019 Publication	<1 %
20	K. Radecka. "The role of super-fast transforms in speeding up quantum computations", Proceedings 32nd IEEE International Symposium on Multiple- Valued Logic ISMVL-02, 2002 Publication	<1 %

21	Submitted to Sim University Student Paper	<1 %
22	Colin P. Williams. "Explorations in Quantum Computing", Springer Science and Business Media LLC, 2011 Publication	<1 %
23	"TM112 Introduction to computing and IT Block 3 isbn9781473022300", Open University Publication	<1 %
24	HAO-YUNG LO, CHIEN-CHUN SU. "A distributive D-algorithm for generating the test pattern for faulty combinational circuit", International Journal of Electronics, 1989 Publication	<1 %
25	Submitted to King's College Student Paper	<1 %
26	Submitted to Lincoln High School Student Paper	<1 %
27	Submitted to Universiti Sains Malaysia Student Paper	<1 %
28	Mondal, B., P. Sarkar, P. K. Saha, and S. Chakraborty. "Synthesis of Balanced Ternary Reversible Logic Circuit", 2013 IEEE 43rd International Symposium on Multiple-Valued Logic, 2013. Publication	<1 %

29	citeseerx.ist.psu.edu Internet Source	<1 %
30	Graduate Texts in Physics, 2015. Publication	<1 %
31	dspace.nwu.ac.za Internet Source	<1 %
32	joems.springeropen.com Internet Source	<1 %
33	Santanu Pattanayak. "Quantum Machine Learning with Python", Springer Science and Business Media LLC, 2021 Publication	<1 %
34	Submitted to University of Edinburgh Student Paper	<1 %
35	baadalsg.inflibnet.ac.in Internet Source	<1 %
36	www.eit.lth.se Internet Source	<1 %
37	Nikhil Saluja. "SAT-based ATPG using multilevel compatible don't-cares", ACM Transactions on Design Automation of Electronic Systems, 4/1/2008 Publication	<1 %
38	quantum-computing.ibm.com Internet Source	<1 %

39	H. Ito, T. Matsubara, T. Kurokawa, Y. Koga. "A proposal of fault-checking fuzzy control", [1992] Proceedings The Twenty-Second International Symposium on Multiple-Valued Logic, 1992 Publication	<1 %
40	Computational Complexity, 2012. Publication	<1 %
41	Dan C. Marinescu. "Preliminaries", Classical and Quantum Information, 2012 Publication	<1 %
42	Submitted to Eastern Mediterranean University Student Paper	<1 %
43	Peikun Wang, Amir Masoud Gharehbaghi, Masahiro Fujita. "Automatic Test Pattern Generation for Double Stuck-at Faults Based on Test Patterns of Single Faults", 20th International Symposium on Quality Electronic Design (ISQED), 2019 Publication	<1 %
44	Diogo Fernandes, Carla Silva, Inês Dutra. "Using Grover's search quantum algorithm to solve Boolean satisfiability problems, part 2", XRDS: Crossroads, The ACM Magazine for Students, 2019 Publication	<1 %

45	www.nap.edu Internet Source	<1 %
46	www.scilit.net Internet Source	<1 %
47	Submitted to BITS, Pilani-Dubai Student Paper	<1 %
48	Casper van der Kerk, Attila Csala, Aeilko H. Zwinderman. "Quantum Computing in the Biomedical Sciences; A Brief Introduction into Concepts and Applications", Computer and Information Science, 2019 Publication	<1 %
49	pdfs.semanticscholar.org Internet Source	<1 %
50	raiith.iith.ac.in Internet Source	<1 %
51	"TM112 Introduction to computing and IT 2 Block 2 isbn9781473022294", Open University Publication	<1 %
52	Submitted to Brunel University Student Paper	<1 %
53	Kampke, T.. "Generating random numbers by adapted rejection", European Journal of Operational Research, 19901016 Publication	<1 %

54	Marius Nagy, Selim G. Akl. "Quantum computation and quantum information†", International Journal of Parallel, Emergent and Distributed Systems, 2006 Publication	<1 %
55	Mosca, Michele. "Quantum Algorithms", Encyclopedia of Complexity and Systems Science, 2009. Publication	<1 %
56	sunsite.icm.edu.pl Internet Source	<1 %
57	Laszlo Gyongyosi, Sandor Imre. "Unsupervised Machine Learning Control of Quantum Gates in Gate-Model Quantum Computers", Frontiers in Optics / Laser Science, 2018 Publication	<1 %
58	Stephen DiAdamo, Marco Ghibaudi, James Cruise. "Distributed Quantum Computing and Network Control for Accelerated VQE", IEEE Transactions on Quantum Engineering, 2021 Publication	<1 %
59	www.geeksforgeeks.org Internet Source	<1 %
60	Submitted to eur Student Paper	<1 %

61	philpapers.org Internet Source	<1 %
62	www.openu.ac.il Internet Source	<1 %
63	Benenti, . "First Experimental Implementations", Principles Of Quantum Computation And Information Volume II Basic Tools and Special Topics, 2007. Publication	<1 %
64	Hsiao-Yu Chiang, Yung-Chih Chen, De-Xuan Ji, Xiang-Min Yang, Chia-Chun Lin, Chun-Yao Wang. "LOOPLock: LOgic OPtimization based Cyclic Logic Locking", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020 Publication	<1 %
65	Magnus Johannesson. "Theory and Methods of Economic Evaluation of Health Care", Springer Nature, 1996 Publication	<1 %
66	Robert Willis, Brad A. Finney. "Environmental Systems Engineering and Economics", Springer Science and Business Media LLC, 2004 Publication	<1 %
67	dspace.cuni.cz Internet Source	<1 %

68	github.com Internet Source	<1 %
69	mafiadoc.com Internet Source	<1 %
70	C T Bhunia. "Quantum Computing & Information Technology: A Tutorial", IETE Journal of Education, 2015 Publication	<1 %
71	Eleanor Rieffel. "An introduction to quantum computing for non-physicists", ACM Computing Surveys, 9/1/2000 Publication	<1 %
72	H. De Raedt, K. Michielsen, A. Hams, S. Miyashita, K. Saito. "Quantum spin dynamics as a model for quantum computer operation", The European Physical Journal B - Condensed Matter, 2002 Publication	<1 %
73	Velev, M.N.. "Effective use of Boolean satisfiability procedures in the formal verification of superscalar and VLIW microprocessors", Journal of Symbolic Computation, 200302 Publication	<1 %
74	arxiv.org Internet Source	<1 %
	futuretpm.eu	

75	Internet Source	<1 %
76	pt.scribd.com Internet Source	<1 %
77	riptutorial.com Internet Source	<1 %
78	www.scribd.com Internet Source	<1 %
79	A.U. Khalid, Z. Zilic, K. Radecka. "FPGA emulation of quantum circuits", IEEE International Conference on Computer Design: VLSI in Computers and Processors, 2004. ICCD 2004. Proceedings., 2004 Publication	<1 %
80	B. P. Lanyon, C. Hempel, D. Nigg, M. Muller et al. "Universal Digital Quantum Simulation with Trapped Ions", Science, 2011 Publication	<1 %
81	Ciaran Hughes, Joshua Isaacson, Anastasia Perry, Ranbel F. Sun, Jessica Turner. "Quantum Computing for the Quantum Curious", Springer Science and Business Media LLC, 2021 Publication	<1 %
82	Jamroz, . "Fundamentals of Interaction of Light with Matter", Optical Science and	<1 %

Engineering, 2006.
Publication

83 Ta Yeong Wu, Ningqun Guo, Chee Yang Teh, Jacqueline Xiao Wen Hay. "Chapter 2 Theory and Fundamentals of Ultrasound", Springer Science and Business Media LLC, 2013 <1%
Publication

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		

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:

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.

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

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

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

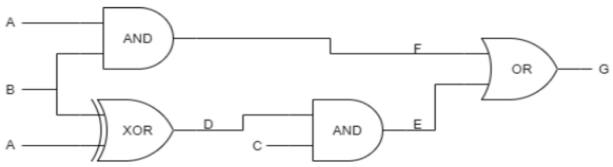


Fig. 1 Circuit diagram of Carry

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

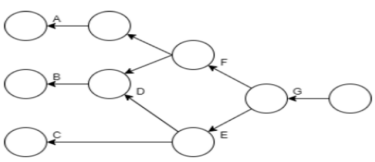


Fig. 2 Directed Acyclic Graph of Carry

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

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.

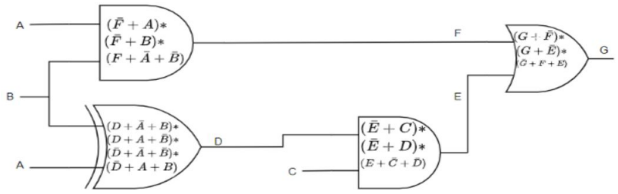


Fig. 3 Carry Circuit with labelled gates

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:

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

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

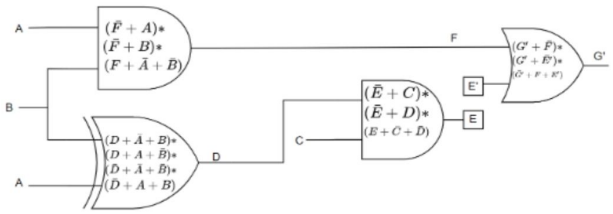


Fig. 4 Carry Circuit with E stuck at 1

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:

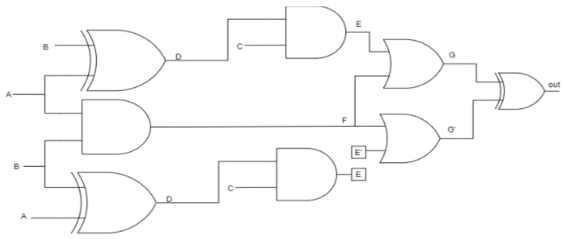


Fig. 5 Final carry circuit with XOR gate whose output is 1

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:

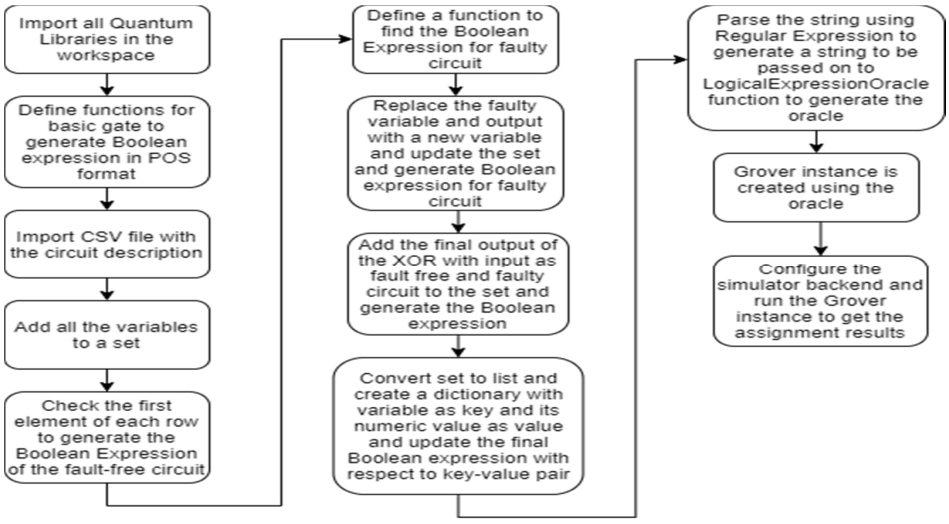


Fig. 6 Flowchart of the algorithm

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.

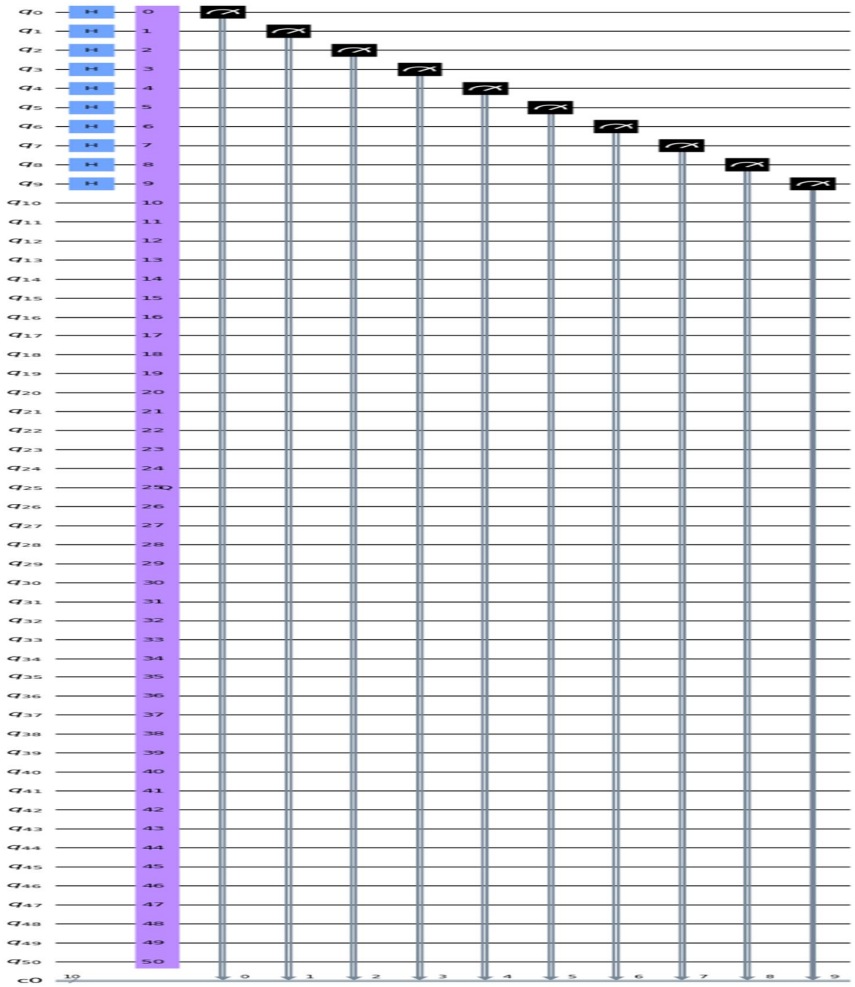


TABLE 2: Number of gates

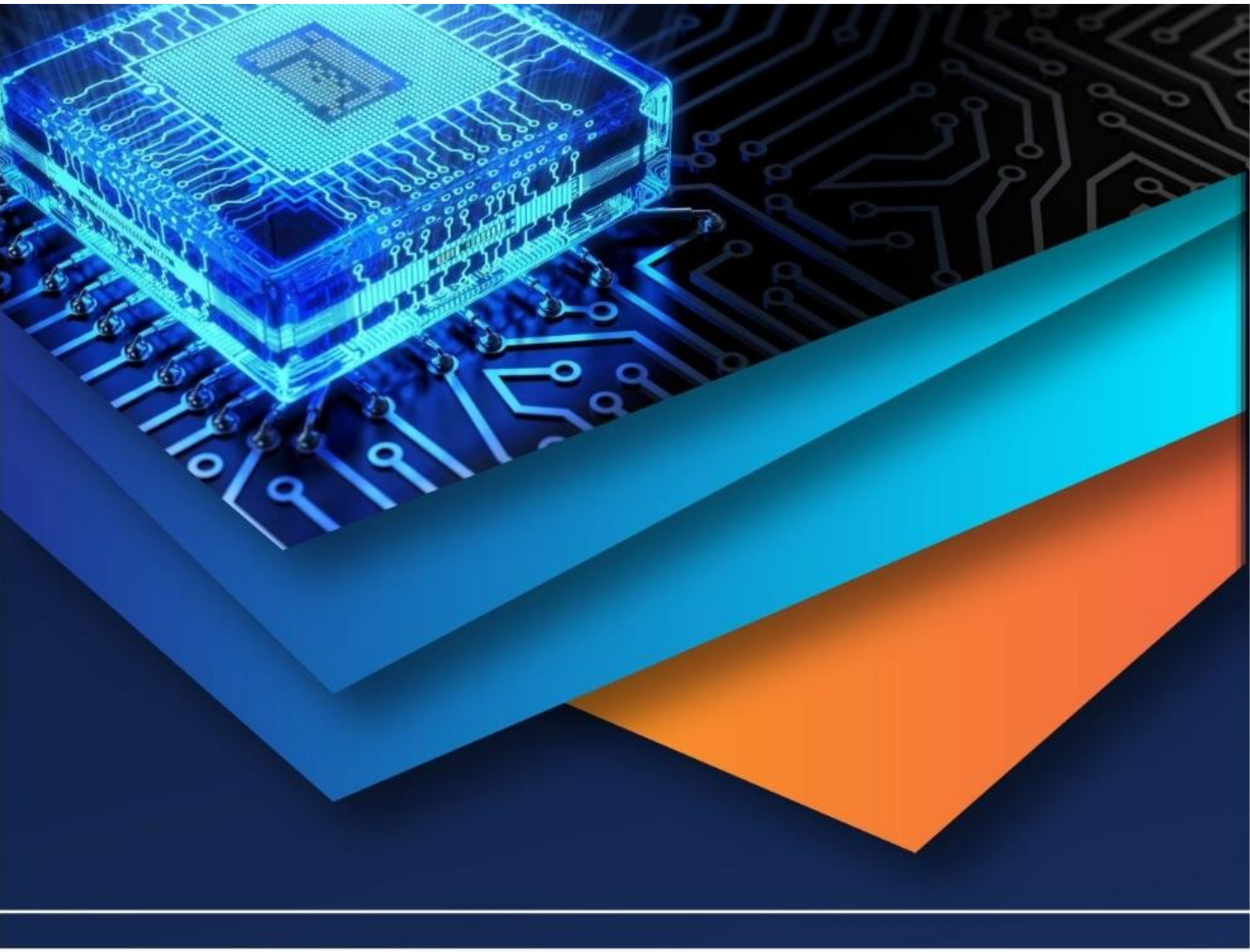
Quantum Gate	Count
cx (controlled x)	2022
u1,u2,u3 (unitary gates)	1695
t (toffoli)	123
x	53
h (hadamard gate)	2

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