A Review Report on

**Exploring the Potential of Quantum Computing**

by

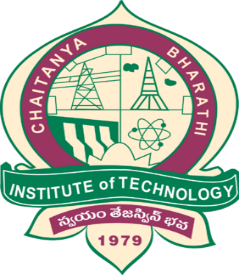
**Ankitha Pilli**

1601157337122

Under the mentorship of

**Dr. K. Sagar**

Professor



Department of Computer Science and Engineering

Chaitanya Bharathi Institute of Technology (Autonomous)

Gandipet, Hyderabad – 500075

January, 2019

**CERTIFICATE**

This is to certify that the seminar review report titled “*Exploring the potential of Quantum Computing*” is a bona-fide work carried out by Ankitha P, in completion of a seminar under my mentorship. The results embodied in this report have not been submitted to any other university or institute for the award of any seminar credits.

Seminar Guide Seminar In-Charge Seminar In-Charge

**Dr. K. Sagar G. Vanitha G. Vivek**

Professor Asst Professor Asst Professor

Department of CSE Department of CSE Department of CSE

**ABSTRACT**

Today’s computers help and entertain us, connect us with people all over the world, and allow us to process huge amounts of data to solve problems and manage complex systems. However, there are problems that today’s systems will never be able to solve. For challenges above a certain size and complexity, we don’t have enough computational power today to tackle them. To stand a chance at solving some of these complex problems, we need a new kind of computing: one whose computational power scales exponentially as the system size grows. Quantum computation is an emerging field of research at the intersection of computer science, information theory and quantum physics. The idea behind quantum computers is to take the phenomena of the quantum realm and use it to our advantage to create better machines. For instance, if a task requires us to find one correct answer out of 100 million choices, an ordinary computer would go through 50 million steps to do so whereas a quantum computer would only go through 10,000. Today, real quantum computers can be accessed through the cloud, and many thousands of people have used them to learn, conduct research, and tackle new problems. Quantum computers can one day provide breakthroughs in many disciplines, including materials and drug discovery, the optimization of complex systems, and artificial intelligence. But to realize those breakthroughs, and to make quantum computers widely useable and accessible, we need to reimagine information processing and the machines that do it.

**TABLE OF CONTENTS**

Certificate

Abstract

1. Introduction

1.1 Motivation 5

1.2 Seminar Problem Definition 5

1.3 Objective of the Report 5

1.4 Prospective Applications 6

2. Literature Survey

2.1 Introduction 7

2.2 Existing System and Features 7

2.2.1 Primitive Methods 8

2.2.2 Contemporary Methods 9

2.3 Major Problems in Existing Systems 9

2.4 Proposed Model 10

2.5 GAN Model Architecture 10

2.6 Advantages of Proposed GAN Model 11

References 13

**LIST OF FIGURES**

Figure 2.1 Generative Adversial Network Architecture 10

Figure 2.2 Latent Vector giving rise to various 3D images 11

**INTRODUCTION**

**1.1. Motivation**

.

**1.2. Problem Definition**

**1.3. Objective of the Project**

The objective of this report is to explain the potential of Quantum computing and how it can benefit the human-kind.

**1.4. Prospective Applications**

Following are the prospective applications of Quantum Computing.

**LITERATURE SURVEY**

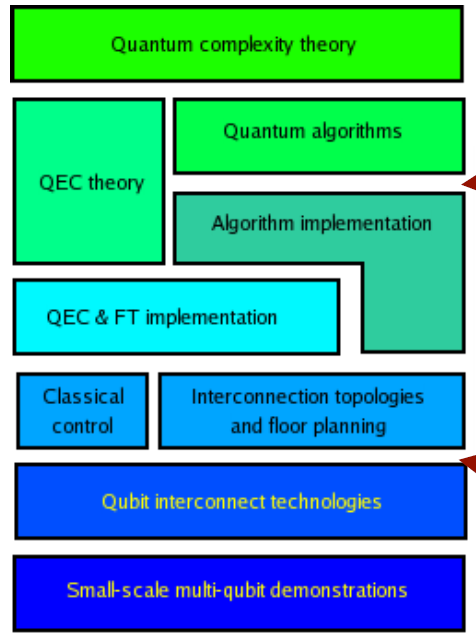
**2.1. Introduction**

**2.2. Existing System and Features**

**2.3. Major Problems in Existing Systems**

**2.4. Proposed Model**

**2.5. Quantum Computer Architecture:**



.

**2.6. Advantages of the Quantum Model**

This project has a lot of scope with regards to future trends in technology and applications that benefit humankind.

In this chapter, the bases of quantum computing are introduced. Section A.1 will first introduce

the mathematical formalism used in quantum mechanics and explain the notions

of ‘quantum bit’, ‘quantum measurement’, ‘superposition’ and ‘entanglement’. Then,

section A.2 will present the theoretical abstraction used in quantum computing and explain

why a theoretical model of computation is crucial. To close this chapter, section

A.3 will explain in more details the theoretical abstraction used in the majority of the

scientific papers: the quantum circuit model.

**A.1 Quantum computing basics**

**A.1.1 What is a quantum bit?**

The most elementary block of classical computing is the bit, an element that can only takes two values: either 0 or 1. Physically, a bit can be implemented in various ways such as magnetic orientation (used in hard drive disks), flip-flop circuits (used in random access memory) or a voltage (used to transmit the information on a cable).

A quantum bit or ‘qubit’ can be seen as an *improved classical bit*: it has all the properties of a classical bit plus some additional properties that a classical bit does not

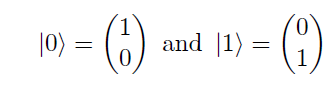
have access to. The two core properties that differentiate a classical bit from a qubit are ‘superposition’ and ‘entanglement’. Before introducing these two notions, we will introduce the mathematical formalism used in the field of quantum computing.

**A.1.3 Superposition states**

A *bit* is defined as a quantity that can have only one of two fixed values. A bit is always

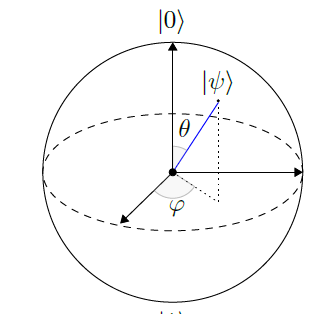
either in one value *or* in the other.

This property does not hold anymore for a qubit: a qubit can choose one value from an infinite set of possible states, called *superposition state. A* single quantum bit, or qubit, has the luxury of an infinite choice of so-called superposition states. Nature allows it [the qubit] to have a part corresponding to **0** *and* a part corresponding to **1** *at the same time*, analogous to the way a musical note contains various harmonic frequencies.



The difference between a classical bit and a quantum bit in terms of attainable states

is clearly visible by using the Bloch sphere representation



The Bloch sphere is the sphere of radius 1, centered at the origin. The possible states for a classical bit are the north and south poles of the sphere (only 0 and 1). The possible states for an isolated quantum bit are all the points located on the surface of the bloch sphere. Each point on the Bloch sphere represents a unique state.

The coefficients are tightly linked with the behaviour of quantum measurement.

The next section is dedicated to this measurement operation.

**A.1.4 Measurement of quantum states**

In order to retrieve the result of an algorithm executed on a quantum computer, we need to be able to read the state of each qubit representing the solution at the end of the algorithm. The only way to read the value stored in qubits is to measure their state and to interpret the result of the measurement. But measurement in the quantum world does not behave as in the macroscopic world. The following sections will introduce the two main characteristics of the measurement operation, only on single-qubit states.

**Measurement is probabilistic**

Let first recall the mathematical representation of a single-qubit state in the basis

The outcome of measuring in the basis *fju⟩ ; jv⟩g* a qubit in the state *j ⟩* will be

probabilistic and the probabilities associated to each outcome are known: *j\_j*2 is the

probability to measure the state *ju⟩* and *j\_j*2 is the probability to measure the state *jv⟩*.

This behaviour is an axiom of quantum mechanics and cannot be derived from physical

principles but was verified in practice in various experiments.

**Measurement destroys superposition**

Taking back the state *j ⟩* in equation A.3, section A.1.4 told us that the outcome of measurement *can* be random. An other important characteristic of quantum measurement is that it collapses the qubit state that is measured into the measured state.

For example imagine we have a qubit in the superposition state *j ⟩* described in. We perform a measurement on this state and the state *ju⟩* is the outcome of the measurement. After the measurement the state *j ⟩* collapsed in *ju⟩* and is no longer in a superposition state (i.e. *j ⟩* = *ju⟩*). Measuring again the state *j ⟩* after the first measurement will then give a deterministic result (*ju⟩* for the previous example) as the qubit is no more in a superposition state.

**A.1.5 Entanglement between qubits**

Entanglement is the second fundamental difference between quantum bits and classical

ones. In order to explain the notion of entangled states we need to define what is a

separable state. A *n*-qubits quantum state is said to be *separable* if it can be written as

the tensor product of the states of its individual components:

*j ⟩* = *j* 1*⟩ j* 2*⟩ \_ \_ \_ j n⟩*

A *n*-qubits quantum state that is not separable is called an *entangled* state. Entangled

states are quantum states that cannot be described only by looking individually at each

of its components. In other words, there is a link between the qubits that cannot be

explained just by looking at each qubit separately.

In order to fully describe *j*\_+*⟩* we should add another condition that will link the qubits

*A* and *B* together. In the case of *j*\_+*⟩* the condition ‘when measured, the states A and

B always output the same result’ is sufficient to suppress the ambiguity: because of the

destructive nature of measurement, measuring qubit *A* will *necessarily* collapse the state

of qubit *A* to either *j*0*⟩* or *j*1*⟩*. But as qubits *A* and *B* are entangled, the state of qubit

*B* will also be impacted by the measurement and depending on the value measured for

*A*, the final state after measuring qubit *A* is either *j*0*⟩ j*0*⟩* or *j*1*⟩ j*1*⟩*.

To summarise, 2 or more qubits are entangled if their states are linked and cannot

be fully described individually.

**A.2 Theoretical background of quantum computing**

Formalising a theoretical background that defines the concept of ‘computation’ in classical

computing has been a major step in computer science. The most famous theoretical

model in classical computing is probably the Turing machine that defines how information

is stored (on an infinite tape) and how ‘computations’ are done (by moving a head

over a cell, reading/writing the cell and moving left or right).

Such an abstraction is fundamental in classical computing as it allows to define the

notions of *computability* and *complexity*. There exist many models of computation in

classical computing, such as the Turing machine, finite state machine, random access

machines, etc.

Before briefly presenting the

existing quantum models of computation we should define the notion of *universality*.

**A.2.1 Model of computation and universality**

A model of computation is said to be *universal* if it can map any input state to any

output state.

Being universal for a classical computer means that it can map any input state

*f*0*;* 1*gn* to any output state *f*0*;* 1*gn* just by using the logical gates allowed by the model

of computation it implements.

Similarly, a quantum model of computation is said to be universal if it can map

any *n*-qubit input state *j n⟩* to any *n*-qubit output state *jϕn⟩*. But as quantum states

are continuous, an other definition of universality is often used in quantum computing.

Most of the time, a quantum model of computation is said to be universal when it can

*approximate arbitrarily close* any output state *jϕn⟩* from any input state *j n⟩*.

The next sections will present some of the quantum models of computation that have

been defined and studied since the creation of quantum computing.

**A.2.2 Measurement based quantum computer**

The measurement based quantum computer, also called the one-way quantum computer,

is a model exclusively based on measurement and its properties.

The qubits composing the quantum computer are initialised in a particular highlyentangled

state called *cluster state* or *graph state*. Once the initial state has been prepared,

computations are performed by measuring qubits in a specific order. Performing

a measurement on one qubit of such an entangled state will, thanks to the properties

of entanglement and measurement, change the state of the qubits that were entangled

with the measured qubit.

An algorithm for such a quantum computer should output a sequence of qubit indexes,

representing the qubits that should be measured. This sequence may depends on

the results obtained on the previous measurements.

It has been proven that measurement based quantum computing is universal, i.e.

it can approximate to an arbitrarily low precision any output state.

**A.2.3 Adiabatic quantum computer**

The adiabatic quantum computer model is based on Hamiltonian simulation and evolution.

It consists in encoding the solution of a problem as the ground state of a given

Hamiltonian and find this ground state.

To find this ground state, the model starts by constructing a simple Hamiltonian

already in its ground state. Then, it evolves this simple Hamiltonian towards the one

related to the solution we are searching for, by keeping the ground state of the evolved

Hamiltonian thanks to the adiabatic theorem. A proof of the equivalence (in term of

complexity) between the quantum circuit model and the adiabatic quantum model can

be found online. This makes the adiabatic quantum computer model a universal model.

**A.2.4 Topological quantum computer**

The topological quantum computer model is based on properties of a fundamental

particle (anyons) that has not been experimentally observed at the moment. The model

is also universal in the sense that it can simulate any quantum circuit with only a polynomial

loss in complexity. Topological qubits are an active subject of research because they would theoretically allow coherence times much larger than qubits based on super-conducting materials. Microsoft is currently investigating topological qubits.

**A.2.5 Quantum circuit**

The quantum circuit model is one of the most used model of computations in quantum

computing today: most of the quantum algorithms are presented by using this model and

nearly all the complexity analysis make use of this model to reason about the number

of needed operations.

**Definition of the model**

The model of quantum circuit defines a computation as a sequence of quantum gates,

which are reversible operations acting on *n*-qubits. In the quantum circuit model, all

the qubits are initialised to the state *j*0*⟩*.

The quantum circuit model is similar to the classical model of computation in two

ways:

1. There are unitary blocks in both models: quantum gates for the quantum model

and logical gates for the classical one.

2. The unitary blocks are chained in an ordered sequence.

But there is also a major difference between the quantum and the classical models:

quantum operations need to be reversible whereas classical ones are not (AND logical

gate for example).

**Quantum circuit representations**

Quantum circuit model is the most used one today in the field of quantum computing. Because there is no unique, standardised, *easily readable* and widely-used quantum language for the moment, quantum circuits and quantum algorithms are represented in a visual way.

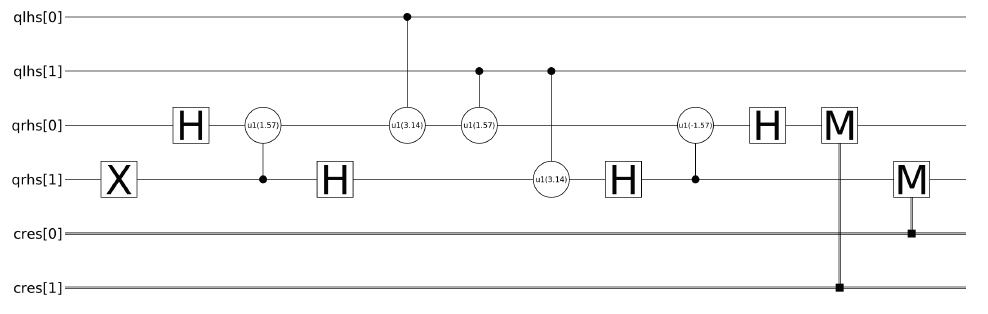


Figure A.3 – Visual representation of a quantum circuit. The quantum circuit is an

in-place 2-qubit adder modulo 3 (i.e. no overflow checks are performed).

Generated with the qasm2image tool

Figure A.3 illustrates how a quantum circuit is represented. The graphical representation

used follows the following rules:

• Time evolution goes from left to right.

• Each **qubit** is represented by an horizontal line. The qubit identifier can be written

at the extremities of the line.

• Each **classical bit** is represented by a doubled horizontal line.

• Each **gate** has a unique representation. Single-qubit gates (like X) and measurement

are represented by a square. Controlled single-qubit gates (CX for example)

use circles.

**A.3 Quantum gates and algorithms**

The previous section described quantum computing basics. This section will explain how qubits can be manipulated in practice to perform quantum computations:

1. Introduction to the mathematical formalism behind the operators acting on qubit states

2. Examples of some quantum operators

The whole section uses the model of *quantum circuit* and *quantum gates*. This model is similar to the one used in classical computing: a quantum circuit (resp. a *program* for classical computing) is composed of quantum gates (resp. logical gates) applied sequentially to a given number of qubits (resp. bits). Thanks to the similarity between

the classical and the quantum models, many mathematical tools and formalisms from

classical computing can be re-used in the quantum world.

In addition, this section will not deal with physical implementation of qubits and quantum gates. To do an analogy with classical computing, this section will explain how works the AND logical gate and how to use it in algorithms without showing the underlying electric circuit implementing it.

**A.3.1 Qubit state transformation**

**Mathematical formalism**

‘Qubit state transformations’ are operations applied on one or several qubit(s) that may

change their state to an other *valid* state. From quantum mechanic properties, these

quantum operations must be linear. This means that a quantum operation *U* can be

represented as a matrix acting on the quantum state space.

The property of quantum operation to map quantum states to quantum states implies

that unit length vectors must remain of unit length vectors, or equivalently that the

operation is *unitary*.

Linearity of quantum operators is enough to prove a fundamental theorem of quantum

programming: the no-cloning principle. This theorem say that a quantum operator *U*

that clones a quantum state does not exists.

The quantum operator presented in this section is often called a *quantum gate*.

**Quantum gates**

Quantum gates are the quantum equivalent of logic gates in classical programming: they

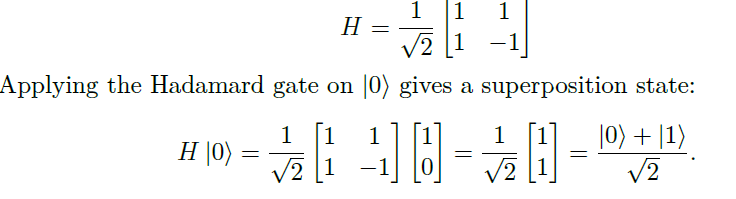
are used as base to construct more advanced quantum operations. Some of the most

used quantum gates and their mathematical definition is given below.

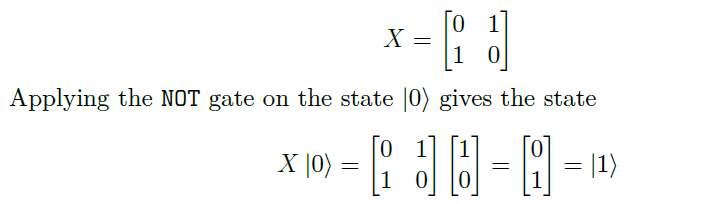
**Hadamard gate** is one of the most used quantum gate in quantum computation.

Hadamard gate acts on a single qubit and is used to create a superposed state from *j*0*⟩*

or *j*1*⟩*.

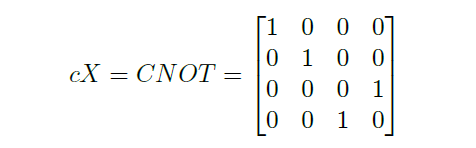
 **Pauli-X gate** is also called NOT gate because of its effects: it flips the state of the

quantum bit.

 **controlled NOT gate** is the most used 2-qubits gate. It takes as input a control qubit

and a target qubit and applies a NOT gate to the target qubit if and only if the control

qubit is in the state *j*1*⟩*.



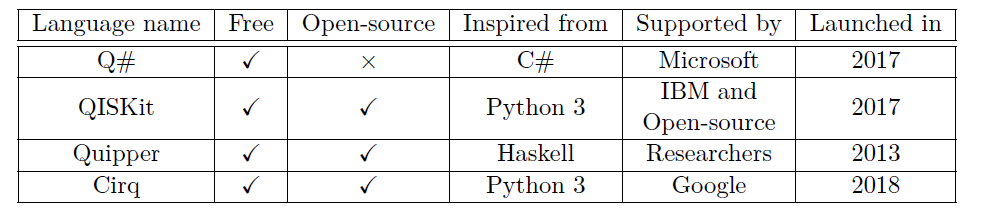
This gate is commonly used to create entanglement between 2 qubits.

More generally, for any quantum gate *U* the quantum gate controlled-*U* or c-*U* is a

gate that applies *U* to the target qubit if the control qubit is in the state *j*1*⟩*.

**Quantum programming languages**

There are plenty of quantum programming languages, Table 2.2 lists the main languages used to produce quantum machine instructions (i.e. a code in quantum assembly).

Table 2.2 – Main quantum programming languages in July 2018. (This list is not exhaustive.)

In Table 2.2, each programming language uses a different paradigm to produce and run quantum code and has a different level of maturity.

• **Q#**, created by Microsoft. It can be integrated into C# or F# code. Q# code works like a black box: the user/developer does not have access to the low-level representation of its code (i.e. the translation of the code in quantum assembly).

Everything is abstracted in order to make Q# programming a ‘high-level’ programming language.

Q# comes with a quantum library developed by Microsoft. This library implements most of the basic quantum algorithms such as the Quantum Fourier Transform or Shor’s algorithm.

• **QISKit**  is an open-source project launched by IBM. The QISKit project is composed of:

**–** The **OpenQASM** language specification.

**–** Several **SDKs** allowing developers to generate OpenQASM code with their favourite programming language. The available languages in July 2018 are Python 3, Swift and JavaScript.

**–** An **API** to submit quantum programs to IBM’s quantum chips that are in the cloud.

The QISKit project provides a way to generate OpenQASM code from Python, Swift and JavaScript.

• **Quipper** is designed to be a scalable programming language for quantum computing. Quipper is based on Haskell and uses the functional paradigm.

• **Cirq** was released in July 2018. It is a Python 3 framework developed by Google that specifically targets Bristlecone, the quantum chip unveiled several months ago by Google.

**Quantum machine learning**

Machine learning is a field of computer science that is currently exploding in terms of research paper published and obtained results. One of the principal problem that machine learning aims at solving is classification problem. Depending on the availability of labelled data, the classification problem has two different formulations:

i. If labelled data is available, then we are performing ‘supervised learning’ and the problem is to be able to classify a given object within a given set of classes that are characterised by the labelled data.

ii. If the data we have is not labelled, then we are performing ‘unsupervised learning’. In this case, the problem is to find the labels that best fit the provided data.

There exist plenty of classical algorithms in both supervised and unsupervised machine learning. The case of quantum algorithms applied to supervised or unsupervised machine learning has been investigated in. In this papers, Lloyd, Mohseni and Rebentrost introduce and analyse a new quantum algorithm that mimics Lloyd’s algorithm (the author of this algorithm *is not* one of the authors of even if the have the same name). The quantum version of Lloyd’s algorithm for finding a local minima of the *k*-means problem has an asymptotic complexity of *O*(*k* log (*kMN*) /*ϵ*) with *k* the number of classes, *M* the number of points in the data set, *N* the dimension of the search space (i.e. the number of coordinates needed to represent one data point) and *ϵ* the desired precision. This complexity can be lowered down to *O*(log (*kMN*) /*ϵ*) if the *k* clusters representing the different classes are ‘relatively well separated’. In comparison, Lloyd’s algorithm on a classical computer has an asymptotic time complexity of *O*(*kMNI*) where *I*, the number of iterations needed, may scale as 2Ω(√*n*) in the worst case.

**REFERENCES**

[1] Solenov, Dmitry et al. “The Potential of Quantum Computing and Machine Learning to Advance Clinical Research and Change the Practice of Medicine”*Missouri medicine* vol. 115,5 (2018): 463-467.

[2] Prathamesh Ratnaparkhi, B. K. Behera, and P. K. Panigrahi, “Getting Started With Quantum Computation: Experiencing The Quantum Experience,” *Unpublished*, 2018. doi: 10.13140/RG.2.2.24079.23207

[3] A. Frisch, "IBM Q — Introduction into quantum computing with live demo," 2017 30th IEEE International System-on-Chip Conference (SOCC), Munich, 2017, pp. 1-2. doi: 10.1109/SOCC.2017.8225985

[4] Suau Adrien, Master’s thesis report on “Introduction to Quantum Computing”, Aug,2018.

[5] Quantum Research Portal of IBM- <https://www.research.ibm.com/ibm-q/learn/what-is->quantum-computing/

[6] NBC news of Quantum Computing- <https://www.nbcnews.com/mach/video/what-is->quantum-computing-1381810755943

# [7] Quantum Computers Explained – Limits of Human Technology

https://www.youtube.com/watch?v=JhHMJCUmq28