

**QUANTUM COMPUTER SIMULATOR BASED ON  
QUTRITS AND THE CIRCUIT MODEL OF QUANTUM  
COMPUTATION**

Md. Shamsul Kaonain  
Student ID: 06101002

**Department of Computer Science and Engineering**  
December 2009

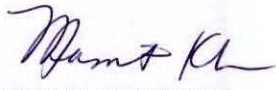


**BRAC University, Dhaka, Bangladesh**

## DECLARATION

I hereby declare that this thesis is based on the results found by myself. Materials of work found by other researcher are mentioned by reference. This Thesis, neither in whole nor in part, has been previously submitted for any degree.

Signature of  
Supervisor



Dr. Mumit Khan

Signature of  
Author



Md. Shamsul Kaonain

## ACKNOWLEDGMENTS

I would like to begin by thanking my thesis supervisor Dr. Mumit Khan for allowing me to work on this thesis under his supervision and for his continuous support and guidance. It was a wonderful experience for me, and I shall remain grateful to him forever.

I would like to thank Dr. Arshad Momen for extending every possible help when asked for, and giving his valuable time to answer my questions even if they were trivial.

I would also like to thank out lecturer Mr. Annajiat Alim Rasel for his suggestions and advice from time to time.

I would also like to thank my family members, especially my parents for their support and belief in me all throughout.

Last but not the least, I would like to thank Almighty Allah for his blessings, without which nothing is possible.

## ABSTRACT

The number of quantum bits that can currently be realized is limited; therefore extending the state space to explore multi-valued options is reasonable. A quantum computer simulator based on qutrits (3 valued quantum bits) and the circuit model of quantum computation is presented along with an implementation of Quantum Fourier Transform using the simulator.

**Keywords:** Quantum Computer, Quantum Computing, Ternary logic,  
Multi valued logic, Quantum Fourier Transform, Quantum Computer simulator



## Table of Contents

DECLARATION.....	ii
ACKNOWLEDGMENTS.....	iii
ABSTRACT.....	iv
Table of Contents.....	1
List of Figures .....	3
List of Tables .....	3
CHAPTER 1: INTRODUCTION .....	4
CHAPTER 2: BASIC CONCEPTS .....	8
2.1 Linear Algebra.....	8
2.1.1 Braket Notation .....	8
2.1.2 Hilbert Space.....	9
2.2 Quantum Mechanics .....	11
2.2.1 State Space.....	11
2.2.2 Evolution .....	12
2.2.3 Measurement .....	12
CHAPTER 3: BASICS OF QUANTUM COMPUTING.....	14
3.1 Basic concepts .....	14
3.2 Quantum Gates.....	17
3.2.1 Single qubit gates.....	17
3.2.2 Two qubit gates .....	20
3.2.3 Three qubit gates.....	23
CHAPTER 4: QUANTUM COMPUTING BASED ON QUTRITS.....	25
4.1 Need for Multivalued quantum bits.....	25
4.2 Qutrits.....	26
4.3 Ternary Quantum Gates.....	27
CHAPTER 5: THE QUANTUM COMPUTER SIMULATOR .....	28

5.1 Development Tools .....	28
5.2 Design .....	28
5.3 Implementing QFT using the simulator.....	29
CHAPTER 6 CONCLUSION .....	31
6.1 Successes and failures .....	31
6.2 Future work .....	31
6.3 Concluding Remarks .....	32
REFERENCES .....	33
APPENDIX A: JAVA CODE FOR THE MAIN SIMULATOR CLASS.....	34
APPENDIX B: SCREEN DUMP AND OUTPUT OF SIMULATOR FOR QFT.....	43



## List of Figures

Fig 1.1 Moore's Law Statistics.....	1
Fig 3.1 Qubit represented by two electronic labels in an atom.....	14
Fig 3.2 Bloch sphere representation of a qubit.....	16
Fig 3.3 Matrices for the Pauli Gates.....	17
Fig 3.4 Circuit symbols used for the Pauli gates.....	18
Fig 3.6 The Hadamard Gate.....	18
Fig 3.7 Visualization of the Hadamard gate on the Bloch sphere, acting on $( 0\rangle +  1\rangle)/\sqrt{2}$ .....	19
Figure 3.8 Circuit symbol for CNOT gate and the matrix representation of the gate.....	20
Fig 3.9 Circuit for swapping two qubits, and the general symbol for the swap gate.....	20
Fig 3.10 Circuit symbol and matrix representation of controlled-Z gate.....	21
Fig 3.11 Circuit symbol and matrix representation of controlled-phase gate.....	21
Fig 3.12 Circuit symbol and matrix representation of the Toffoli gate.....	22
Fig 3.13 Circuit symbol and matrix representation of the Fredkin gate.....	23
Fig 4.1 Exponential increase of state space with multi-valued logic.....	25
Fig 4.2 Matrix representation of the Chrestenson Gate.....	26
Fig 5.1 Pseudo code for the simulator.....	27
Fig 5.2 Implementation of QFT using ternary gates.....	28

## List of Tables

Table 2.1 Commonly used Expression in Dirac Notation.....	8
---	---

## CHAPTER 1: INTRODUCTION

After the invention of the transistor in 1947, rapid development of computer hardware began. Computer hardware has grown in power at an amazing pace ever since, so much that the growth was codified by Gordon Moore in 1965, in what has come to be known as Moore's Law.

Moore's law describes a long-term trend in the history of computing hardware. Since the invention of the integrated circuit in 1958, the number of transistors that can be placed inexpensively on an integrated circuit has increased exponentially, doubling approximately every two years. The trend was first observed by Intel co-founder Gordon E. Moore in a 1965 paper. It has continued for almost half a century and in 2005 was not expected to stop for at least another decade.

Amazingly enough, Moore's law has approximately held true in the decades since the 1960s. Nevertheless, most observers expect that this dream run will end sometime during the first two decades of the twenty-first century.

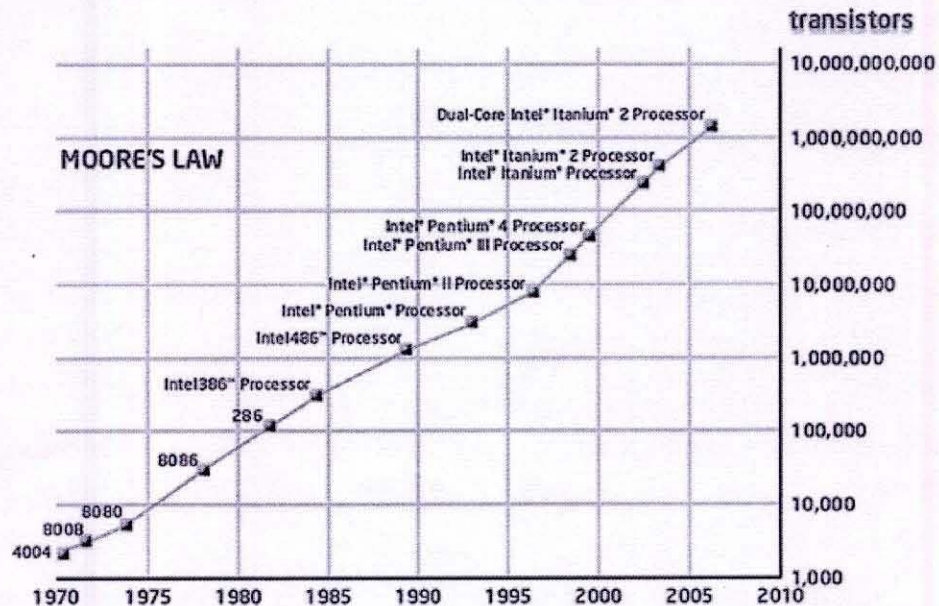


Fig 1.1 Moore's Law Statistics



Current methods of fabrication of computer chips are beginning to run up against fundamental difficulties of size. As integrated circuits and electronic devices are made smaller and smaller, Quantum effects are beginning to interfere with the functioning of such circuits and devices.

One of the possible solutions to the problem posed by eventual failure of devices in a sub atomic scale is to move to a different architecture of computing. One option is provided by the theory of quantum computation, which is based on the idea of carrying out computation using quantum mechanics as opposed to conventional computing which uses the laws of classical physics.

At the turn on the 20th century, classical physics gave rise to a series of crises, by predicting absurdities such as the existence of "ultra violet catastrophe" involving infinite energies, or electrons spiraling inexorably into the atomic nucleus. At first such problems were resolved by addition of ad hoc hypotheses to classical physics, but as we gained better understanding of atoms and radiation, these attempted explanations became more and more convoluted. The crisis came to a head in the early 1920s with the creation of the modern theory of Quantum Mechanics.

Quantum Mechanics is a mathematical framework or set of rules for the construction of physical theories. For example, there is a physical theory known as quantum electrodynamics which describes with fantastic accuracy the interactions of atoms and light. Quantum Electrodynamics is built up within the framework of quantum mechanics, but it contains specific rules not determined by quantum mechanics. The relationship of Quantum mechanics to specific physical theories like quantum electrodynamics is rather like the relationship of a computer's operating system to specific applications software – the operating system sets certain basic parameters and modes of operation, but leaves open how specific tasks are accomplished by the applications.



In early 1980s, interest arose in whether it might be possible to use quantum effects to send signals faster than light, a big no-no according to Einstein's theory of relativity. The resolution of this problem turns out to hinge on whether it is possible to clone an unknown quantum state, that is, construct a copy of a quantum state. Cloning turns out not to be possible in general in quantum mechanics.

This no cloning theorem, discovered in the early 1980s, is one of the earliest results in this field. Many refinements of this theorem have since been developed, and now we have conceptual tools which allow us to understand how well a (necessarily imperfect) quantum cloning device would work.

A major 1970 development is the complete control over single quantum systems. Since the 1970s many techniques for controlling single quantum systems have been developed. For example, methods have been developed for trapping a single atom in 'atom trap', isolating it from the rest of the world and allowing us to probe many different aspects of its behavior with incredible precision. The scanning tunneling microscope has been used to move single atoms around, creating designer arrays of atoms at will. Electronic devices whose operation involves the transfer of only single electrons have been demonstrated. By obtaining complete control over single quantum systems, we are controlling untouched regions of nature in the hopes of discovering new and unexpected phenomena.

Quantum Computation and Quantum Information fit naturally into this program. They provide useful series of challenges at varied levels of difficulty for people devising methods to better manipulate single quantum systems, stimulate the development of new experimental techniques and provide the most interesting directions in which to take the experiment. Conversely, the ability to control single quantum systems is essential if we are to harness the power of quantum mechanics for applications to Quantum Computing and Quantum Information.



Ordinary computers cannot simulate a quantum computer efficiently, thus we can conclude that quantum computers would offer us an essential speed advantage over classical computers. This advantage is so significant that many researchers are of the opinion that any amount of progress in classical computation would fail to bridge the gap between the powers of a classical computer and that of a quantum computer.

In spite of this intense interest, there has been modest success to date in the development of Quantum Computers. Small quantum computers, capable of doing dozens of operations on few qubits represent state of the art in Quantum Computation [1].

This paper is organized as follows. Chapter 2 reviews basics of linear algebra and quantum mechanics followed by the basics of Quantum Computing in chapter 3 which includes description of basic quantum gates based on a 2 dimensional Hilbert space (see chapter 2.1.2 for definition) and the circuit model of quantum computing. Chapter 4 deals with ternary versions of quantum gates followed by a description of the quantum computer simulator in Chapter 5.

## CHAPTER 2: BASIC CONCEPTS

### 2.1 Linear Algebra

#### 2.1.1 Bracket Notation

The “braket” notation is a very compact formalism for linear algebra which was introduced by Paul Dirac. It was introduced by Dirac in order to describe in a uniform manner vectors and linear operators both in the abstract Hilbert space<sup>1</sup>  $\mathcal{H}$ . This is the standard notation of quantum mechanics for linear algebraic concepts, so Braket notation will be used throughout the paper. Table 2.1 lists the most commonly used expressions.

Table 2.1

Commonly used Expression in Dirac Notation

Notation	Description
$ \psi\rangle$	general “ket” vector, e.g. $ \psi\rangle = (c_0, c_1, \dots)^T$
$\langle\psi $	dual “bra” vector to $ \psi\rangle$ e.g. $\langle\psi  = (c_0^*, c_1^*, \dots)$
$ n\rangle$	$n^{\text{th}}$ basis vector of some standard basis $N = \{ 0\rangle,  1\rangle, \dots\}$
$ \tilde{n}\rangle$	basis vector of an alternate basis $\tilde{N} = \{ \tilde{0}\rangle,  \tilde{1}\rangle, \dots\}$
$\langle\phi \psi\rangle$	inner product of $ \psi\rangle$ and $ \phi\rangle$
$ \phi\rangle \otimes  \psi\rangle$	tensor product of $ \psi\rangle$ and $ \phi\rangle$
$ \phi\rangle \psi\rangle$	abbreviated tensor product $ \phi\rangle \otimes  \psi\rangle$
$ i, j\rangle$	abbreviated tensor product of the basis vectors $ i\rangle$ and $ j\rangle$
$M^\dagger$	adjoint operator (matrix) $M^\dagger = (M^T)^*$
$\langle\phi M \psi\rangle$	inner product of $ \phi\rangle$ and $M \psi\rangle$
$\ \psi\ $	abbreviated norm $\ \psi\ $



### 2.1.2 Hilbert Space

All quantum mechanical states have an associated complex Hilbert space associated with it. Following are definitions that are required to define Hilbert space followed by the definition of Hilbert Space.

**Definition 2.1.2.1** A set  $\mathbf{V}$  is called vector space over a scalar field  $\mathbf{F}$  iff the operations  $+$ :  $\mathbf{V} \times \mathbf{V} \rightarrow \mathbf{V}$  (vector addition) and  $\cdot$ :  $\mathbf{F} \times \mathbf{V} \rightarrow \mathbf{V}$  (scalar multiplication) are defined, and

- (i)  $(\mathbf{V}, +)$  is a commutative group,
- (ii)  $\lambda|\psi\rangle = |\psi\rangle\lambda$ ,
- (iii)  $\lambda(\mu|\psi\rangle) = (\lambda\mu)|\psi\rangle$ ,
- (iv)  $(\lambda + \mu)|\psi\rangle = \lambda|\psi\rangle + \mu|\psi\rangle$ ,
- (v)  $\lambda(|\psi\rangle + |\phi\rangle) = \lambda|\psi\rangle + \lambda|\phi\rangle$ .

All vector spaces considered here are complex vector spaces, i.e.  $\mathbf{F} = \mathbf{C}$ .

**Definition 2.1.2.2** Let  $\mathbf{V}$  be a complex vector space. A function  $\langle \cdot | \cdot \rangle : \mathbf{V} \times \mathbf{V} \rightarrow \mathbf{C}$  is called inner product iff

- (i)  $\langle \psi | (\lambda|\phi\rangle + \mu|\chi\rangle) \rangle = \lambda\langle \psi | \phi \rangle + \mu\langle \psi | \chi \rangle$ ,
- (ii)  $\langle \psi | \phi \rangle = \langle \phi | \psi \rangle^*$ ,
- (iii)  $\langle \psi | \psi \rangle \in \mathbf{R}$ ,  $\langle \psi | \psi \rangle \geq 0$ ,  $\langle \psi | \psi \rangle = 0 \Leftrightarrow |\psi\rangle = \mathbf{o}$ .

An inner product also defines the norm  $\| |\psi\rangle \| = \sqrt{\langle \psi | \psi \rangle}$  (also written as  $\| \psi \|$ ).

The following inequalities apply:

$$|\langle \psi | \phi \rangle| \leq \| \psi \| \| \phi \| \quad (\text{Schwarz inequality}) \quad (2.1)$$

$$\| |\psi\rangle + |\phi\rangle \| \leq \| \psi \| + \| \phi \| \quad (\text{triangle inequality}) \quad (2.2)$$

**Definition 2.1.2.3 (Completeness)** Let  $\mathbf{V}$  be a vector space with the norm  $\|\cdot\|$  and  $|\psi_n\rangle \in \mathbf{V}$  a sequence of vectors.

(i)  $|\psi_n\rangle$  is a Cauchy sequence iff  $\forall \epsilon > 0 \exists N > 0$  such that

$$\forall n, m > N, \|\psi_n - \psi_m\| < \epsilon \quad (2.3)$$

(ii)  $|\psi_n\rangle$  is convergent iff there is a  $|\psi\rangle \in \mathbf{V}$  such that

$$\forall \epsilon > 0 \exists N > 0 \forall n > N, \|\psi_n - \psi\| < \epsilon \quad (2.4)$$

**Definition 2.1.2.4** A complete vector space  $\mathcal{H}$  with an inner product  $\langle \cdot | \cdot \rangle$  and the corresponding norm  $\|\psi\| = \sqrt{\langle \psi | \psi \rangle}$  is called Hilbert space.

A Hilbert space  $\mathcal{H}$  is *separable* if there exists an enumerable set  $S \subseteq \mathcal{H}$  which is dense in  $\mathcal{H}$ , i.e. for any  $|\psi\rangle \in \mathcal{H}$  and  $\epsilon > 0$  there exists a  $|\sigma\rangle \in S$  with  $\|\psi - \sigma\| < \epsilon$ .

Separable Hilbert spaces are considered when dealing with Quantum Computing.

### 2.1.3 Linear Operators

**Definition 2.1.3.1** Let  $\mathbf{V}$  be a vector space and  $A$  be a function  $A: \mathbf{V} \rightarrow \mathbf{V}$ .  $A$  is called linear operator on  $\mathbf{V}$  iff

$$A(\lambda|\psi\rangle + \mu|\phi\rangle) = \lambda A(|\psi\rangle) + \mu A(|\phi\rangle) = \lambda A|\psi\rangle + \mu A|\phi\rangle \quad (2.5)$$

In  $\mathbf{C}^n$ , a linear operator  $A$  can be written as a  $n \times n$  matrix with the matrix elements  $a_{ij} = \langle i | A | j \rangle$

$$A = \begin{pmatrix} a_{0,0} & \cdots & a_{0,n-1} \\ \vdots & \ddots & \vdots \\ a_{n-1,0} & \cdots & a_{n-1,n-1} \end{pmatrix} = \sum_{i,j} a_{ij} |i\rangle \langle j| \quad (2.6)$$



Because of (2.5), a linear operator on a vector space  $\mathbf{V}$  with the basis  $\mathbf{B}$  is completely defined by its effect on the basic vectors, so the above operator  $A$  could also be written as

$$A : |n\rangle \rightarrow \sum_k a_{kn} |k\rangle \quad \text{with } |k\rangle \in B \quad (2.7)$$

**Definition 2.1.3.2** The operator  $A^\dagger = (A^T)^* = \sum_{i,j} a_{ji}^* |i\rangle\langle j|$  is called adjoint operator of  $A$ .

Suppose  $A$  is any linear operator on a Hilbert space,  $V$ . It turns out that there exists a unique linear operator  $A^\dagger$  on  $V$  such that for all vectors  $|v\rangle, |w\rangle \in V$ ,

$$(|v\rangle, A|w\rangle) = (A^\dagger|v\rangle, |w\rangle) \quad (2.8)$$

As already shown above, this linear operator is known as the adjoint or *Hermitian conjugate* of the operator  $A$ .

An operator  $A$  whose adjoint is equal to itself is known as a *Hermitian* or *self-adjoint* operator.

## 2.2 Quantum Mechanics

**2.2.1 Postulate 1(State Space)** Associated to any isolated physical system is a complex vector space with inner product (that is, a Hilbert space) known as the *state space* of the system. The system is completely described by its *state vector*, which is a unit vector in the system's space.

How the state space of a given physical system is constructed is beyond the scope of this postulate.

In Quantum Computing, the state of a quantum bit or qubit is the simplest non-trivial quantum mechanical system with a state space  $\mathbf{B} = \mathbf{C}^2$ . The state of the qubit can be described by a linear combination of the basis states.

**2.2.2 Postulate 2 (Evolution)** The evolution of a *closed* quantum system is described by a *unitary transformation*. If  $|\psi\rangle$  is the state of a system in time  $t_1$  is related to the state  $|\psi'\rangle$  of the system at time  $t_2$  by a unitary operator  $U$  which depends only on the times  $t_1$  and  $t_2$ ,

$$|\psi'\rangle = U|\psi\rangle \quad (2.9)$$

Just as quantum mechanics does not tell us the state space or quantum state of a particular quantum system, it does not tell us which unitary operators  $U$  describe real world quantum dynamics. Quantum mechanics merely assures us that the evolution of any closed system can be described in this manner. An obvious question that may come to mind is: which unitary operators would need to be considered? In the case of single qubits, it turns out that any unitary operator can be realized in realistic systems.

**2.2.3 Postulate 3 (Measurement)** Quantum measurements are described by a collection  $\{M_m\}$  of *measurement operators*. These are operators acting on the state space of the system being measured. The index  $m$  refers to the measurement outcomes that may occur in the experiment. If the state  $|\psi\rangle$  of the quantum system is immediately before the measurement then the probability that result  $m$  occurs is given by

$$p(m) = \langle \psi | M_m^\dagger M_m | \psi \rangle, \quad (2.10)$$

and the state of the system after the measurement is

$$\frac{M_m |\psi\rangle}{\sqrt{\langle \psi | M_m^\dagger M_m | \psi \rangle}} \quad (2.11)$$



The measurement operators satisfy the *completeness equation*,

$$\sum_m M_m^\dagger M_m = I \quad (2.12)$$

The completeness equation expresses the fact that probabilities sum to one:

$$1 = \sum_m p(m) = \sum_m \langle \psi | M_m^\dagger M_m | \psi \rangle \quad (2.13)$$

## CHAPTER 3: BASICS OF QUANTUM COMPUTING

### 3.1 Basic concepts

The fundamental concept of Quantum computation is the “quantum bit” or “qubit”, which is analogous to the classical bit. Qubits can be thought to be mathematical objects with certain specific properties.

Like bits, qubits are physically realizable. However, treating qubits as abstract entities give us freedom to construct a general theory for Quantum Computation and Quantum Information which does not depend upon a specific system for its realization.

Like the bit, the qubit also has states. The state space of a qubit is the Hilbert space  $\mathbf{B} = \mathbf{C}^2$ . Two possible states for a qubit are  $|0\rangle$  and  $|1\rangle$ . Here “ $|\ \rangle$ ” is the Dirac notation. The orthonormal system  $\{|0\rangle, |1\rangle\}$  is called *computational basis*.

The states  $|0\rangle$  and  $|1\rangle$  are analogous to the classical bit states 0 and 1. The fundamental difference between the bit and the qubit is that the qubit can be in a state other than  $|0\rangle$  or  $|1\rangle$ . It is also possible for them to exist as a linear combination of the basis states ( $|0\rangle$  and  $|1\rangle$ ), often called superpositions:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle. \quad (3.1)$$

The numbers  $\alpha$  and  $\beta$  are complex numbers. The state of a qubit is a vector in a two-dimensional complex vector space. As stated above, the special states  $|0\rangle$  and  $|1\rangle$  are known as the computation basis states, and they form an orthonormal basis for this vector space.

Unlike a classical computer, a qubit cannot be measured to give information about its quantum state, i.e. the values of  $\alpha$  and  $\beta$ . When we measure the qubit, we get either the result 0 with probability  $|\alpha|^2$ , or the result 1



with probability  $|\beta|^2$ . Since total probability is 1,  $|\alpha|^2 + |\beta|^2 = 1$ . Thus in general, a qubit's state is unit vector in a two-dimensional complex vector space.

A qubit, as we saw already, can be in a superposition state, and this is counter to our intuition or "common sense" understanding of the world around us. The classical bit can be thought of as a coin, which when you throw, can yield either a head or a tail, assuming it's an ideal coin, and doesn't balance on its edge. On the other hand, a qubit can exist in a continuum of states between  $|0\rangle$  and  $|1\rangle$  until it is observed. However, when measured, it can only yield either 0 or 1 probabilistically. For example, a qubit can be in the state

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \quad (3.2)$$

which when measured gives the result 0 fifty percent ( $(|1/\sqrt{2}|^2)$ ) of the time, and the result 1 fifty percent of the time. This state is sometimes denoted  $|+\rangle$ .

A qubit can be realized as two different polarizations of a photon, as the alignment of nuclear spin in a uniform magnetic field, or as the two states of an electron orbiting a single atom such as shown in figure 3.1.

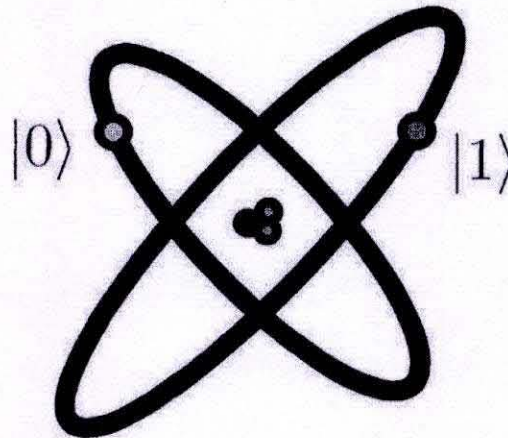


Fig 3.1 Qubit represented by two electronic labels in an atom

In the atom model, the electron can exist in the “ground” or “excited” states, which we will call  $|0\rangle$  and  $|1\rangle$  respectively. By shining light on the atom, with appropriate energy and for an appropriate length of time, it is possible to move the electron from state  $|0\rangle$  to the state  $|1\rangle$  and vice versa. By reducing the time we shine the light, and electron initially in the state  $|0\rangle$  can be moved halfway between  $|0\rangle$  and  $|1\rangle$ , that is the  $|+\rangle$  state.

A geometric representation for qubits exists, which is useful for thinking about qubits. Because  $|\alpha|^2 + |\beta|^2 = 1$ , we may rewrite equation 3.1 as

$$|\psi\rangle = e^{i\gamma} \left( \cos \frac{\theta}{2} |0\rangle + e^{i\varphi} \sin \frac{\theta}{2} |1\rangle \right) \quad (3.3)$$

where  $\theta$ ,  $\varphi$  and  $\gamma$  are real numbers. We can ignore the factor of  $e^{i\gamma}$  in the front because it has no *observable effects*, and for that reason we can effectively write

$$|\psi\rangle = \cos \frac{\theta}{2} |0\rangle + e^{i\varphi} \sin \frac{\theta}{2} |1\rangle. \quad (3.4)$$

The numbers  $\theta$  and  $\varphi$  define a point on the three dimensional sphere as shown in figure 3.2. This sphere called the *Bloch sphere* and it provides a useful means of visualizing the state of a single qubit and thus serves as an excellent tool for ideas about quantum computation and quantum information.



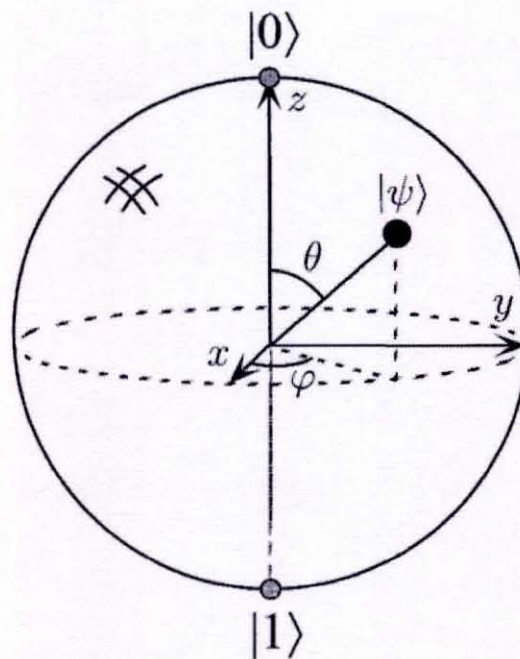


Fig 3.2 Bloch sphere representation of a qubit

### 3.2 Quantum Gates

#### 3.2.1 Single qubit gates

Classical computer circuits are composed of *wires* and *logic gates*. The wires are required to carry information around the circuit, and the logic gates are used to manipulate the information to convert it from one form to another. The only non-trivial member of classical single bit logic gates is the NOT gate, whose operation is defined by its *truth table*, in which  $0 \rightarrow 1$  and  $1 \rightarrow 0$ , that is the states 0 and 1 are interchanged.

The quantum analogous to the classical NOT gate acts *linearly*, that is, it takes the state

$$\alpha|0\rangle + \beta|1\rangle \quad (3.5)$$

to the corresponding state in which the roles of  $|0\rangle$  and  $|1\rangle$  has been interchanged.

$$\alpha|1\rangle + \beta|0\rangle \quad (3.6)$$

Why the quantum NOT gate acts linearly and not in a nonlinear fashion is not obvious right away. It turns out that linear behavior is a general property of quantum mechanics. Using a matrix  $X$  to represent the quantum NOT gate, we can express the NOT gate as:

$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad (3.7)$$

Therefore, quantum gates on a single qubit can be described by two by two matrices. Normalization requires  $|\alpha|^2 + |\beta|^2 = 1$  for a quantum state  $\alpha|0\rangle + \beta|1\rangle$

This must also be true for the resulting state a gate has been applied. It turns out that the appropriate condition on the matrix representing the gate is that the matrix  $U$  describing the single qubit gate be *unitary*, that is the matrix  $U$  multiplied by its *adjoint* or *Hermitian conjugate* would give the identity matrix  $I$ .

Amazingly, this is the only constraint on quantum gates. Any unitary matrix is a valid quantum gate. In contrast to the classical case, where only one non-trivial single bit gate exists, there are many non-trivial single qubit gates.

The common single qubit gates are Pauli gates, Phase and  $\pi/8$  gates, and the Hadamard Gate. Symbols and matrices for the gates are given below.

$$X \equiv \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y \equiv \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z \equiv \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Fig 3.3 Matrices for the Pauli Gates



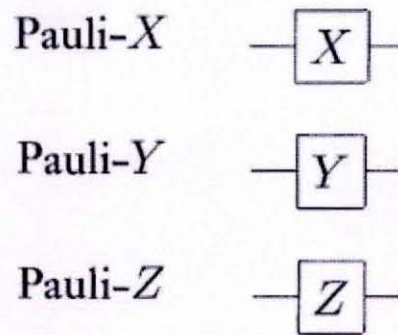


Fig 3.4 Circuit symbols used for the Pauli gates

Of these, the Z gate leaves  $|0\rangle$  unchanged, and flips the sign of  $|1\rangle$  to give  $-|1\rangle$ .

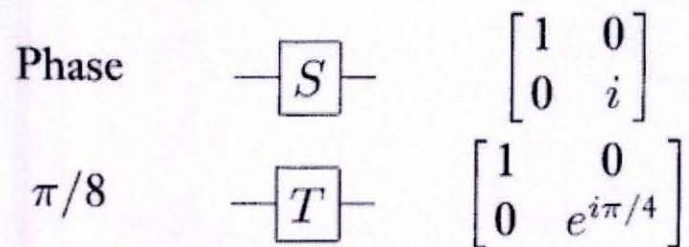
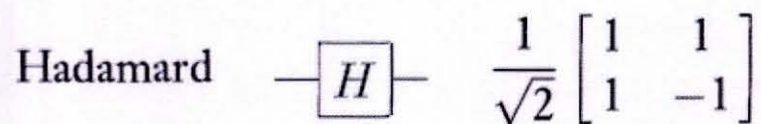
Fig 3.5 Matrices and circuit symbols for the Phase and  $\pi/8$  gates

Fig 3.6 The Hadamard Gate

The Hadamard gate is sometimes described as the “square root of NOT” gate, in that it turns a  $|0\rangle$  into  $(|0\rangle + |1\rangle)/\sqrt{2}$ , that is halfway between  $|0\rangle$  and  $|1\rangle$ , and turns  $|1\rangle$  into  $(|0\rangle - |1\rangle)/\sqrt{2}$ , which is also halfway between  $|0\rangle$  and  $|1\rangle$ .

The Hadamard gate is one of the most useful quantum gates and it is worth trying to visualize its operation by considering the Bloch sphere picture given below:

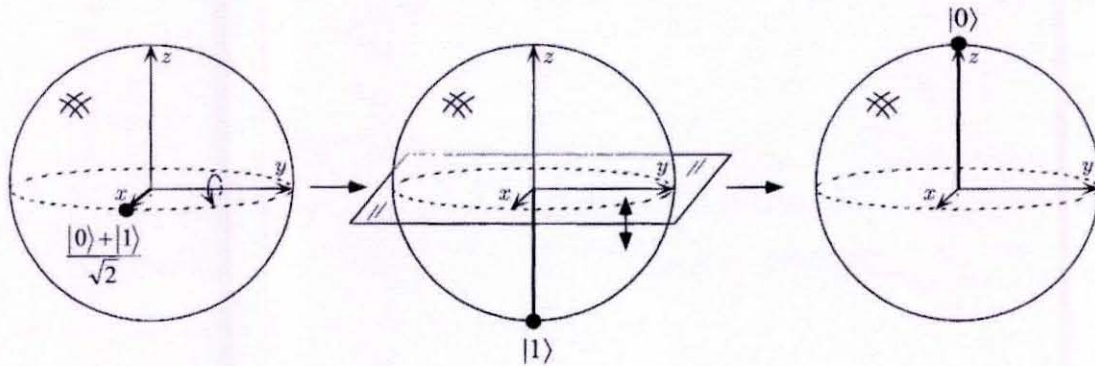


Fig 3.7 Visualization of the Hadamard gate on the Bloch sphere, acting on input state  $(|0\rangle + |1\rangle)/\sqrt{2}$

It turns out that single qubit gates correspond to rotations and reflections of the sphere in this picture. The Hadamard operation is just a rotation of the sphere about the  $y$  axis by 90 degrees, followed by a reflection through the  $x$ - $y$  plane, as shown in figure 3.7.

### 3.2.2 Two qubit gates

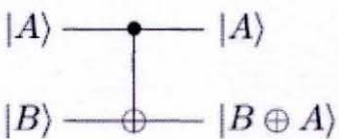
The prototypical multi-qubit quantum logic gate is the *controlled*-NOT or CNOT gate. This gate has two input qubits, known as the *control* qubit and *target* qubit, respectively. If the control qubit is set to 0, then the target qubit passes the gate unchanged. If the control qubit is set to 1, then the target qubit is flipped. The following equations illustrate the action of the CNOT gate:

$$|00\rangle \rightarrow |00\rangle; |01\rangle \rightarrow |01\rangle; |10\rangle \rightarrow |11\rangle; |11\rangle \rightarrow |10\rangle. \quad (3.8)$$

One way of describing the CNOT gate is as a generalization of the classical XOR gate. We can summarize the action of the gate as  $|A, B\rangle \rightarrow |A, B \oplus A\rangle$  where  $\oplus$  is addition in modulo two, which is what the XOR gate does.

Another way of describing the action of the CNOT gate is via its matrix representation which is given in the figure 3.8, along with the circuit symbol used to represent CNOT gates.

controlled-NOT



$$U_{CN} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Figure 3.8 Circuit symbol for CNOT gate and the matrix representation of the gate.

Other useful two qubit gates include swap gates, controlled-Z and controlled-phase gates.

Swap gates, as the name suggests accomplished the simple but useful task of swapping the state of two qubits. A swap gate can be achieved as a combination of CNOT gates, and shown in the figure below.

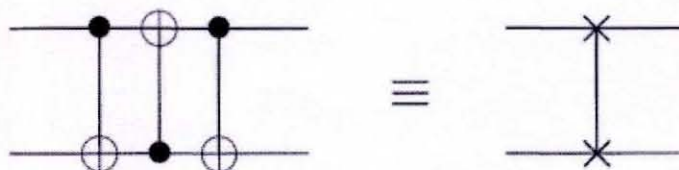


Fig 3.9 Circuit for swapping two qubits, and the general symbol for the swap gate



It may be useful to mention here that unlike classical circuits, "loops", i.e. feedback from one of the circuit to another is not allowed in quantum circuits. Quantum circuits are acyclic. Classical circuits allow wires to be joined together, an operation known as FANIN. This operation is obviously not reversible and therefore not unitary, and is not allowed in quantum circuits. Also, the inverse of this operation, FANOUT, where several copies of a bit are produced from a single wire but is not allowed in quantum circuits. As mentioned earlier in the text, quantum mechanics forbids the copying of a qubit, making the FANOUT operation impossible.

The matrix and circuit symbols for the controlled-Z and controlled-phase gates are given in Figures 3.10 and 3.11, which are controlled versions of the Pauli-Z and phase gates respectively. For any unitary matrix  $U$ , we can have a controlled- $U$  gate, where the value of the control qubit decides whether the input qubits pass unmodified or the gate  $U$  is applied.



Fig 3.10 Circuit symbol and matrix representation of controlled-Z gate



Fig 3.11 Circuit symbol and matrix representation of controlled-phase gate

### 3.2.3 Three qubit gates

Is it possible to simulate classical logic circuits using quantum circuits? The answer to this question is yes. The answer is surprising because classical elements are inherently irreversible whereas quantum elements are reversible, so one would expect that we cannot directly simulate classical logic circuits using quantum circuits. Using a three qubit gate known as *Toffoli* gate, it is possible to make an equivalent quantum circuit to replace any classical circuit.

The Toffoli gate has three input bits and three output bits. Two of the bits are *control bits* that are unaffected by the action of the Toffoli gate. The third bit is the *target* bit that is flipped if both control bits are set to 1, and otherwise is left alone. Applying the Toffoli gate twice to a set of bits brings them back to their initial state, and hence, the Toffoli gate is a reversible gate because it has an inverse, itself!

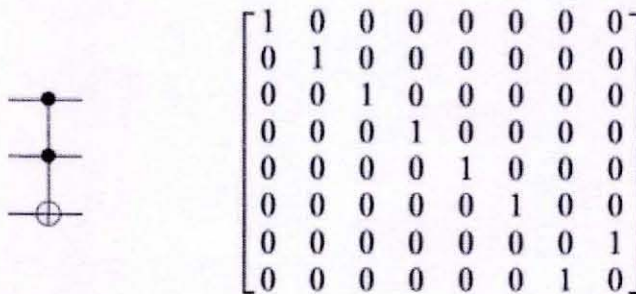


Fig 3.12 Circuit symbol and matrix representation of the Toffoli gate

Another three qubit gate that needs discussion is known as the *Fredkin* gate. The properties of this gate can be used to get an informative overview about general characteristics of reversible logic gates and circuits. It has three input qubits and three output qubits, out of which the third bit is the control bit, which remains unchanged by the action of the Fredkin gate. If the control qubit is 0, the two *target* qubits pass the gate unchanged. However, if the control qubit is set to 1, the states of the *target* qubits are swapped.



An interesting property of the Fredkin gate is the number of 1s is conserved between the input and output qubits. Such conservative properties are of interest to physicists because they can be motivated by fundamental physical principles. The Fredkin gate is a universal gate as well, i.e. it can be cascaded to simulate any classical circuit. Figure 3.13 shows the circuit symbol and matrix representation for the Fredkin gate.

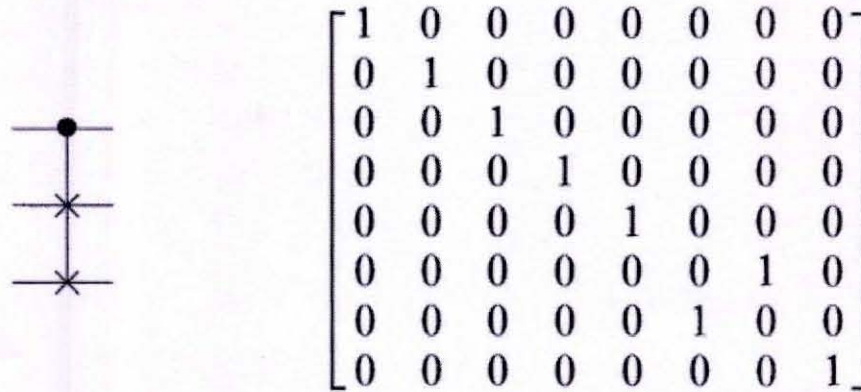


Fig 3.13 Circuit symbol and matrix representation of the Fredkin gate



## CHAPTER 4: QUANTUM COMPUTING BASED ON QUTRITS

### 4.1 Need for Multivalued quantum bits

The main obstacle in quantum computing in current times is posed by the inability to realize practical quantum systems operating on a large number of qubits [3] [4].

It is extremely difficult to isolate a multi particle quantum system, and interaction with the environment would make the system lose its quantum coherence, that is the states of the quantum bits would change due to interaction with the environment, a process known as *decoherence*.

Due to these problems, currently realizable quantum computers can only be of a small number of qubits and researchers are thinking about using multiple valued quantum bits.

Using multiple-valued instead of binary logic has a significant impact on the representable state space and the computational power of quantum machines. It is also worth noting that some known quantum circuit technologies, such as ion traps or quantum dots, are actually three-valued (ternary), rather than binary [3] [4].

The graph in figure 4.1 illustrates the exponential increase in state space as we move from binary to multi-valued logic.

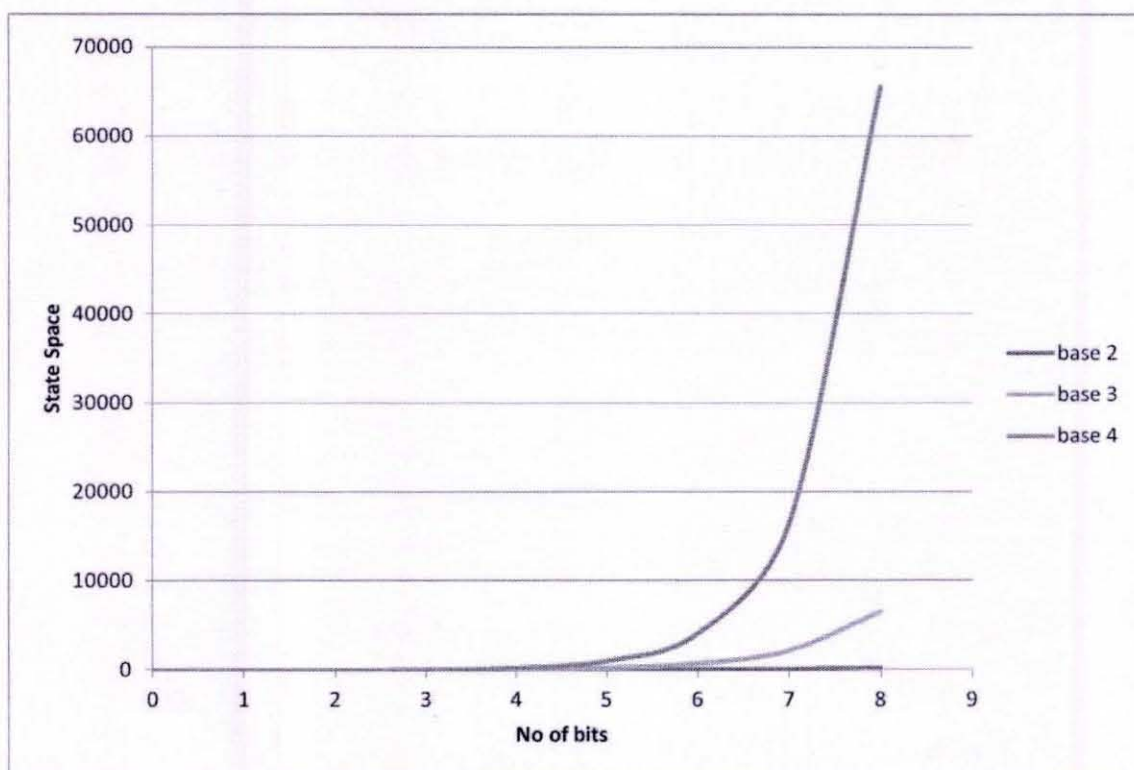


Fig 4.1 Exponential increase of state space with multi-valued logic

## 4.2 Qutrits

**Definition 4.2.1** A  $q$ -ary quantum digit is a multiple valued logic system over basis  $\{|0\rangle, |1\rangle, \dots, |q-1\rangle\}$ .

The ternary quantum digit is often referred to as a *qutrit* in literature, and this notation is adapted in this paper. The state space is a three dimensional Hilbert space. The basis states of a qutrit are  $\{|0\rangle, |1\rangle, |2\rangle\}$ .

Like its binary counterpart, the state of a qutrit is expressed as a linear combination of the basis states as shown below

$$c_0 |0\rangle + c_1 |1\rangle + c_2 |2\rangle \quad (4.1)$$

where  $c_0$ ,  $c_1$ , and  $c_2$  are complex numbers.

### 4.3 Ternary Quantum Gates

Any 3 x 3 unitary matrix specifies a valid 1 qutrit ternary quantum gate. Modulo 3 operations are required for qutrits.

Although the  $q$ -valued quantum states ( $q > 2$ ) come naturally in many cases [5], there exist a few explicit constructions of  $q$ -valued gates [4]. Of these, only the ternary version of the Walsh Hadamard gate and the controlled phase shift gate is considered here.

The ternary version of the Hadamard gate was constructed from the generalized Walsh Hadamard Gate, called as Chrestenson gate after the Chrestenson transform [4] [6].

The Chrestenson gate performs the mapping as given by figure 4.2.

$$CH = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & a & a^2 \\ 1 & a^2 & a \end{bmatrix}.$$

Fig 4.2 Matrix representation of the Chrestenson Gate

Multivalued quantum gates used in 3-valued implementations include a controlled phase shift gate, which applies to the incoming signal the multiplication by a factor  $e^{(i\alpha x)}$ , where  $x$  is the input controlling the amount of shift. Unlike the binary case, there are three possible amounts by which the signal is phase shifted [4]. If the value of  $x$  is 0, then there is no change in the input. If it is 1, then the input qutrits are multiplied by a factor of  $e^{(i\alpha x)}$ , and if it's 2, then the multiplication factor is twice the factor  $e^{(i\alpha x)}$ .



## CHAPTER 5: THE QUANTUM COMPUTER SIMULATOR

### 5.1 Development Tools

- NetBeans IDE 6.7.1 (Build 200907230233)
  
- JDK 1.6.0\_17; Java HotSpot(TM) Client VM 14.3-b01

### 5.2 Design

The complex matrix class developed by Micheal Thomas Flanagan was used (<http://www.ee.ucl.ac.uk/~mflanaga/java/>).

The simulator provides 5 qutrits on which at present only Chrestenson gates and controlled shift gates can be applied. Up to 5 stages of ternary gates can be applied to the 5 qutrits.

A simple pseudo code of the quantum simulator is given below:

- (1) Start
- (2) For each qutrit,
  - 2.1 Check base state selected (NOP,0,1,2)
  - 2.2 Set qutrit to appropriate base state.
- (3) For each qutrit,
  - 3.1 Check Quantum gate selected (CH or Controlled Shift)
  - 3.2 Apply selected gate to qutrit
- (4) For each qutrit
  - 4.1 Output probabilities for each base state.

Fig 5.1 Pseudo code for the simulator

Complete code of the QuReg class used to implement the quantum register and the gates is given in Appendix 1 (page).

### 5.3 Implementing QFT using the simulator

An implementation of QFT (Quantum Fourier Transform) using ternary gates is given in [4], as shown in figure 5.2, by applying Chrestenson gate once to each qubit, followed by the  $p$   $x_k$  ( $k < l$ )e, controlled by :

$$\exp\left(\frac{-2\pi i x_k}{3^{l-k+1}}\right) \quad (5.1)$$

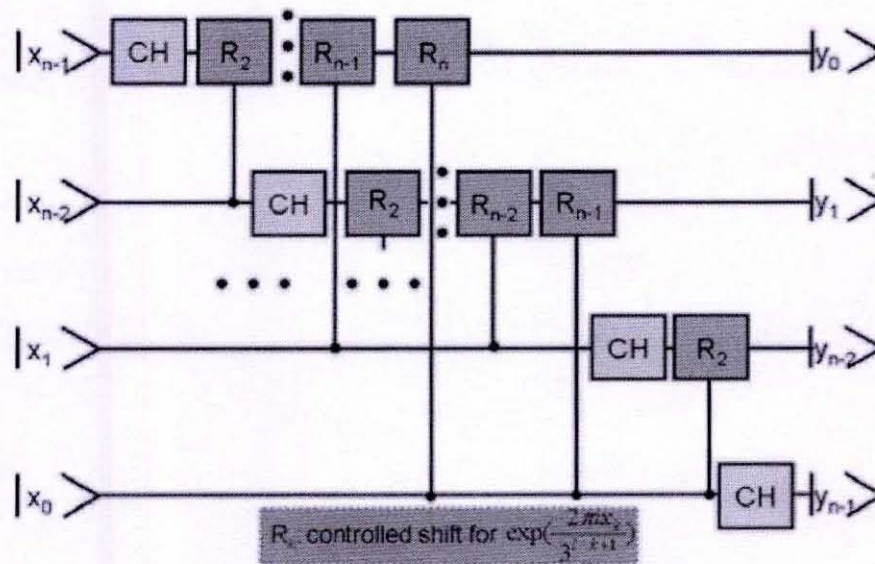


Fig 5.2 Implementation of QFT using ternary gates

The phase shift gates for the implementation of QFT were calculated using (5.1) as given below:

Qutrit 1: No phase shift applied. ( $l = 0, k = 0$ )

Qutrit 2:  $\exp[-2\pi i/3^2]$  with Qutrit 1 used as control. ( $l = 1, k = 0, \text{ for } k < l$ )

Qutrit 3:  $\exp[-2\pi i/3^2]$  with Qutrit 2 used as control and  $\exp[-2\pi i/3^3]$  with Qubit 1 as control. ( $l = 2, k = \{0,1\}$ , for  $k < l$ )

Qutrit 4:  $\exp[-2\pi i/3^2]$  with Qutrit 3 used as control,  $\exp[-2\pi i/3^3]$  with Qutrit 2 as control and  $\exp[-2\pi i/3^4]$  with Qutrit 1 as control. ( $l = 3, k = \{0,1,2\}$ , for  $k < l$ )

Qutrit 5:  $\exp[-2\pi i/3^2]$  with Qutrit 4 used as control,  $\exp[-2\pi i/3^3]$  with Qutrit 3 used as control,  $\exp[-2\pi i/3^4]$  with Qutrit 2 used as control and  $\exp[-2\pi i/3^5]$  with Qutrit 1 used as control. ( $l = 4, k = \{0,1,2,3\}$ , for  $k < l$ )

For the simulator, the first Qutrit is Qutrit 1; in (5.1) the first qutrit is  $x_0$ , which accounts for the differences between the control qutrit obtained by putting in the values of  $l$  and  $k$  and the actual qutrit used in the simulator.

The complete output of the QFT implementation in the simulator and the screen dump of the GUI is given in Appendix B (page).



## CHAPTER 6 CONCLUSION

### 6.1 Successes and failures

The quantum computer simulator was used to implement Chrestenson gates, where the outputs from the simulator match with the expected output. However, there are still small precision errors in output. For e.g. if the Chrestenson gate is applied twice to the state 0 of a qutrit, the expected output is state 0 with a probability of 100%. However, results from the simulator indicate the probability of State 0 to be slightly over 100%, while that of the other two states come to values in the order of  $10^{-31}$  which can be taken to be approximately equal to zero at this stage.

For the phase shift gates however, the multiplicative factor appears to be very small and only changes in the order of  $10^{-9}$  or  $10^{-10}$  is seen in the probabilities which is not acceptable. It is possible that the interpretation of the phase shift gate as shown in chapter 5.3 is erroneous and requires further study to rectify.

### 6.2 Future work

A lot of work needs to be done with the simulator. Only two gates have been implemented so far. Ternary versions of Toffoli gates, Muthukrishnan Shroud gates and Feynman Gates are available, and a lot others has been proposed. The simulator needs to be upgraded to incorporate all such gates.

Entanglement has not been considered, that is in the present design there is no way to know whether a state is entangled or not. One of the major challenges in Quantum Computation is due to entanglement, i.e. determining whether a state is entangled or not, quantification of the entanglement, i.e. calculating the amount of entanglement, and controlling and predicting entanglement. Therefore, the simulator needs to be upgraded so that it can handle entanglement.

Graphical output of the probability distribution would also be helpful for visualizing the output of the simulator. Increasing the number of qutrits that can be handled by the simulator would also be a major improvement.

A web version of the simulator can also be developed that would make it easier for other people to access and use it.

### **6.3 Concluding Remarks**

Multi-valued quantum logic has enormous potential in boosting the power of Quantum algorithms and Quantum computers. For a lot of people, the luster of Quantum Computing seemed to become dim because the promised exponential speedup could be shown only in factoring algorithms which do not have much use outside cryptography.

However, researchers at MIT proposed a quantum algorithm that gives exponential advantage in the solution of linear systems of equations, whose solution is crucial to image processing, video processing, signal processing, robot control, weather modeling, genetic analysis and population analysis, to name just a few applications [7].

Physicists at the National Institute of Standards and Technology (NIST) in USA have demonstrated the first "universal" programmable quantum information processor able to run any program allowed by quantum mechanics using two qubits. The processor could be a module in a future quantum computer, which theoretically could solve some important problems that are intractable today [8].

Also, search giant Google talks about using Quantum algorithms in Machine Learning via adiabatic quantum computers in their research blog.

If such developments continue, Quantum Computers will become a practical reality sooner than we think, and transform the computational capabilities of mankind into the unthinkable.



## REFERENCES

- [1] M.L. Nielsen and I.L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, 2000.
- [2] B. Ömer, *Structured Quantum Programming*, PhD thesis, Technical University of Vienna, 2003.  
URL: <http://tph.tuwien.ac.at/~oemer/doc/qcldoc.pdf>
- [3] X.Deng, T.Hanyu, and M. Kameyama, "Quantum Device Model Based Super Pass Gate for Multiple-Valued Digital Systems", *Proc. Int'l Symp. Multiple-Valued Logic*, pp. 130-138, 1995
- [4] Z. Zilic and K. Radecka, "Scaling and Better Approximating Quantum Fourier Transform by Higher Radices", *IEEE TRANSCATIONS ON COMPUTERS*, VOL. 56, NO.2, pp. 202-207, February 2007
- [5] A. Muthukrishnan and C. Stroud Jr., "Quantum Fast Fourier Transform Using Multilevel Atoms", *J. Modern Optics*, vol. 49, pp. 2115-2127, 2002.
- [6] S.L. Hurst, D.M. Miller, and J. Muzio, *Spectral Techniques in Digital Logic*, Academic Press, 1985.
- [7] Aram W. Harrow, Avinatan Hassidimyand, Seth Lloydz, "Quantum algorithm for linear systems of equations", *Phys. Rev. Lett*, vol. 15, no. 103, pp. 150502 (2009) DOI: 10.1103/PhysRevLett.103.150502
- [8] D. Hanneke, J.P. Home, J.D. Jost, J.M. Amini, D. Leibfried & D.J. Wineland, "Realization of a programmable two-qubit quantum processor", *Nature Physics*, 2009

**APPENDIX A: JAVA CODE FOR THE MAIN SIMULATOR  
CLASS**



```
package simulator;
import flanagan.complex.*;
import flanagan.math.*;

/**
 *
 * @author Md. Shamsul Kaonain
 */
public class QuReg {
    ComplexMatrix register = new ComplexMatrix(1,3) ;

    //To set the qubit to 0,1,2 or NOP

    public void setBase(int base)
    {
        if (base==0)
        {
            this.register.setElement(0,0,1.0,0);
            this.register.setElement(0,1,0,0);
            this.register.setElement(0,2,0,0);
        }

        else if (base==1)
        {
            register.setElement(0,0,0,0);
            register.setElement(0,1,1.0,0);
            register.setElement(0,2,0,0);
        }

        else if (base==2)
        {
            register.setElement(0,0,0,0);
            register.setElement(0,1,0,0);
        }
    }
}
```

```
    register.setElement(0,2,1.0,0);
}

else
{
    register.setElement(0,0,0,0);
    register.setElement(0,1,0,0);
    register.setElement(0,2,0,0);
}
}

//Code to print out the base values
public String PrintBase()
{
    Complex test;
    String out;

    test = register.getElementCopy(0,0);
    out = "(" + test.getReal() + " + i" + test.getImag()+ " , ";

    test = register.getElementCopy(0,1);
    out = out + (test.getReal()+ " + i" + test.getImag()+ " , ");

    test = register.getElementCopy(0,2);
    out = out + (test.getReal()+ " + i" + test.getImag()+ ")";

    return out;
}
```



```
// Code to measure the state probabilities.
public String Measure()
{
    String output;
    Complex test;
    double test1;

    test = register.getElementCopy(0,0);
    test1 =Fmath.truncate((test.squareAbs()*100),5);
    output = "|0> -> " + test1 + "%" + " , ";

    test = register.getElementCopy(0,1);
    test1 =Fmath.truncate((test.squareAbs()*100),5);
    output = output + "|1> -> " + test1 + "%" + " , ";

    test = register.getElementCopy(0,2);
    test1 =Fmath.truncate((test.squareAbs()*100),5);
    output = output + "|2> -> " + test1 + "%";

    return output;

}
```

**//Code for implementing the Chrestenson Gate**

```
public void CHgate()
{
    ComplexMatrix Cgate = new ComplexMatrix(3,3);
    Complex col1,col2,col3,old1,old2,old3;

    Cgate.setElement(0,0,(1/Math.sqrt(3)),0);
    Cgate.setElement(0,1,(1/Math.sqrt(3)),0);
    Cgate.setElement(0,2,(1/Math.sqrt(3)),0);
    Cgate.setElement(1,0,(1/Math.sqrt(3)),0);
    Cgate.setElement(1,1,((Math.cos((-2*Math.PI)/3)/Math.sqrt(3.0)),((Math.sin((-2*Math.PI)/3))/Math.sqrt(3.0)));
    Cgate.setElement(1,2,((Math.cos((-4*Math.PI)/3)/Math.sqrt(3.0)),((Math.sin((-4*Math.PI)/3))/Math.sqrt(3.0)));
    Cgate.setElement(2,0,(1/Math.sqrt(3)),0);
    Cgate.setElement(2,1,((Math.cos((-4*Math.PI)/3)/Math.sqrt(3.0)),((Math.sin((-4*Math.PI)/3))/Math.sqrt(3.0)));
    Cgate.setElement(2,2,((Math.cos((-2*Math.PI)/3)/Math.sqrt(3.0)),((Math.sin((-2*Math.PI)/3))/Math.sqrt(3.0)));

    old1 =register.getElementCopy(0,0);
    old2 =register.getElementCopy(0,1);
    old3 =register.getElementCopy(0,2);

    col1 =old1.times((Cgate.getElementCopy(0,0)));
    col1.plusEquals(old2.times((Cgate.getElementCopy(1,0))));
    col1.plusEquals(old3.times((Cgate.getElementCopy(2,0))));

    col2 = old1.times((Cgate.getElementCopy(0,1)));
    col2.plusEquals(old2.times((Cgate.getElementCopy(1,1))));
    col2.plusEquals(old3.times((Cgate.getElementCopy(2,1))));
```



```
col3 = old1.times((Cgate.getElementCopy(0,2)));
col3.plusEquals(old2.times((Cgate.getElementCopy(1,2))));
col3.plusEquals(old3.times((Cgate.getElementCopy(2,2))));

register.setElement(0,1,col2);
register.setElement(0,0,col1);
register.setElement(0,2,col3);
}

public int getRandomArbitrary(double min, double max)
{
    return (int)((Math.random() * (max - min)) + min);
}

// To determine the truth value of the control qubit for use with the shift gate.
public int TruthValue()
{

    double prob1, prob2, prob3;
    Complex test;
    int Tvalue;

    Tvalue = 0;

    test = register.getElementCopy(0,0);
    prob1 = test.squareAbs()*100;

    test = register.getElementCopy(0,1);
    prob2 =test.squareAbs()*100;

    test = register.getElementCopy(0,2);
```

```
prob3 = test.squareAbs()*100;

if ((prob1==prob2) && (prob2==prob3))
{
    //Random function required
    Tvalue = getRandomArbitrary(0,3);
}
else
{
    if (prob1>prob2)
    {
        if (prob1>prob3)
        {
            //Prob1 is the largest
            Tvalue=0;
        }
        else if (prob1<prob3)
        {
            //Prob3 is the largest
            Tvalue=2;
        }
        else
        {
            Tvalue = getRandomArbitrary(0,3);
            while (Tvalue==1)
            {
                Tvalue = getRandomArbitrary(0,3);
            }
        }
    }
    else if (prob1<prob2)
    {
```



```
if (prob2>prob3)
{
    //Prob2 is the largest
    Tvalue=1;
}
else if (prob2<prob3)
{
    //Prob3 is the largest
    Tvalue=2;
}
else
{
    //Random fuction required
    Tvalue = getRandomArbitrary(0,3);
    while (Tvalue==0)
    {
        Tvalue = getRandomArbitrary(0,3);
    }
}
}
return Tvalue;
}
```

```
//Code to implement the controlled phase shift gate
```

```
public void CPS(int control, int power)
{
    double value;
    Complex Multiplier,old1,old2,old3;

    value = (-2*Math.PI)/(3^power);
    value = value*control;

    Multiplier = this.register.getElementCopy(0, 0);
    Multiplier.setReal(Math.sin(value));
    Multiplier.setImag(Math.cos(value));

    old1 = this.register.getElementCopy(0,0);
    old2 = this.register.getElementCopy(0,1);
    old3 = this.register.getElementCopy(0,2);

    old1.times(Multiplier);
    old2.times(Multiplier);
    old3.times(Multiplier);

    this.register.setElement(0,0,old1);
    this.register.setElement(0,1,old2);
    this.register.setElement(0,2,old3);
}
}
```



**APPENDIX B: SCREEN DUMP AND OUTPUT OF  
SIMULATOR FOR QFT**

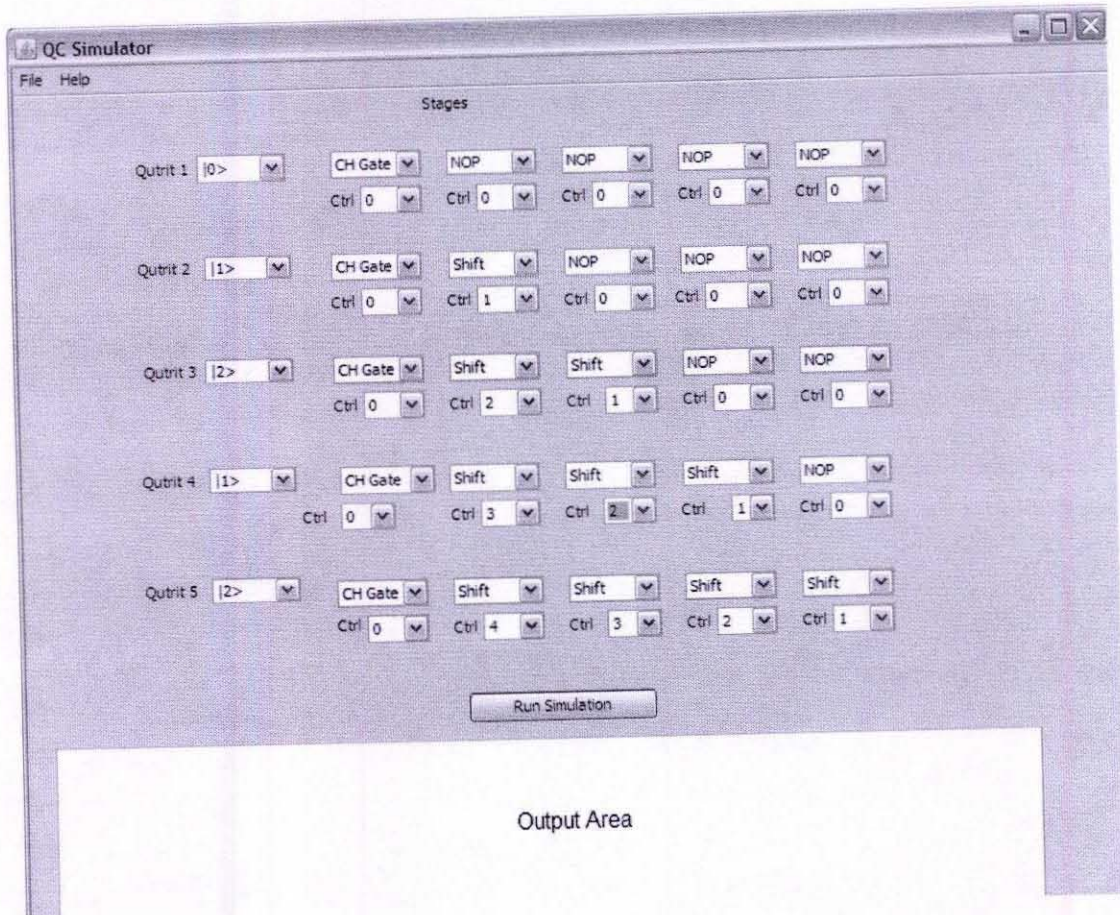


Fig A.1 Screen dump of QFT simulation



## Full output of QFT simulation

```

Qutrit 1 set to |0>
|0> -> 100.0%, |1> -> 0.0%, |2> -> 0.0%
Qutrit 2 set to |1>
|0> -> 0.0%, |1> -> 100.0%, |2> -> 0.0%
Qutrit 3 set to |2>
|0> -> 0.0%, |1> -> 0.0%, |2> -> 100.0%
Qutrit 4 set to |1>
|0> -> 0.0%, |1> -> 100.0%, |2> -> 0.0%
Qutrit 5 set to |2>
|0> -> 0.0%, |1> -> 0.0%, |2> -> 100.0%
Starting Stage 1
Applying CH Gate to Qutrit 1
(0.5773502691896258 + i0.0, 0.5773502691896258 + i0.0, 0.5773502691896258 + i0.0)
|0> -> 33.33333%, |1> -> 33.33333%, |2> -> 33.33333%
Applying CH Gate to Qutrit 2
(0.5773502691896258 + i0.0, -0.28867513459481275 + i-0.5000000000000001, -
0.28867513459481314 + i0.49999999999999994)
|0> -> 33.33333%, |1> -> 33.33333%, |2> -> 33.33333%
Applying CH Gate to Qutrit 3
(0.5773502691896258 + i0.0, -0.28867513459481314 + i0.49999999999999994, -
0.28867513459481275 + i-0.5000000000000001)
|0> -> 33.33333%, |1> -> 33.33333%, |2> -> 33.33333%
Applying CH Gate to Qutrit 4
(0.5773502691896258 + i0.0, -0.28867513459481275 + i-0.5000000000000001, -
0.28867513459481314 + i0.49999999999999994)
|0> -> 33.33333%, |1> -> 33.33333%, |2> -> 33.33333%
Applying CH Gate to Qutrit 5
(0.5773502691896258 + i0.0, -0.28867513459481314 + i0.49999999999999994, -
0.28867513459481275 + i-0.5000000000000001)
|0> -> 33.33333%, |1> -> 33.33333%, |2> -> 33.33333%
Starting Stage 2

```

Applying Shift Gate to Qutrit 2 with Qutrit 1 as control

The truth value is 2

$(0.5773502691896258 + i0.0, -0.28867513459481275 + i-0.5000000000000001, -$   
 $0.28867513459481314 + i0.49999999999999994)$

$|0\rangle \rightarrow 33.33333\%, |1\rangle \rightarrow 33.33333\%, |2\rangle \rightarrow 33.33333\%$

Applying Shift Gate to Qutrit 3 with Qutrit 2 as control

$(0.5773502691896258 + i0.0, -0.28867513459481314 + i0.49999999999999994, -$   
 $0.28867513459481275 + i-0.5000000000000001)$

$|0\rangle \rightarrow 33.33333\%, |1\rangle \rightarrow 33.33333\%, |2\rangle \rightarrow 33.33333\%$

Applying Shift Gate to Qutrit 4 with Qutrit 3 as control

$(0.5773502691896258 + i0.0, -0.28867513459481275 + i-0.5000000000000001, -$   
 $0.28867513459481314 + i0.49999999999999994)$

$|0\rangle \rightarrow 33.33333\%, |1\rangle \rightarrow 33.33333\%, |2\rangle \rightarrow 33.33333\%$

Applying Shift Gate to Qutrit 5 with Qutrit 4 as control

$(0.5773502691896258 + i0.0, -0.28867513459481314 + i0.49999999999999994, -$   
 $0.28867513459481275 + i-0.5000000000000001)$

$|0\rangle \rightarrow 33.33333\%, |1\rangle \rightarrow 33.33333\%, |2\rangle \rightarrow 33.33333\%$

Starting Stage 3

Applying Shift Gate to Qutrit 3 with Qutrit 1 as control

$(0.5773502691896258 + i0.0, -0.28867513459481314 + i0.49999999999999994, -$   
 $0.28867513459481275 + i-0.5000000000000001)$

$|0\rangle \rightarrow 33.33333\%, |1\rangle \rightarrow 33.33333\%, |2\rangle \rightarrow 33.33333\%$

Applying Shift Gate to Qutrit 4 with Qutrit 2 as control

$(0.5773502691896258 + i0.0, -0.28867513459481275 + i-0.5000000000000001, -$   
 $0.28867513459481314 + i0.49999999999999994)$

$|0\rangle \rightarrow 33.33333\%, |1\rangle \rightarrow 33.33333\%, |2\rangle \rightarrow 33.33333\%$

Applying Shift Gate to Qutrit 5 with Qutrit 3 as control

$(0.5773502691896258 + i0.0, -0.28867513459481314 + i0.49999999999999994, -$   
 $0.28867513459481275 + i-0.5000000000000001)$

$|0\rangle \rightarrow 33.33333\%, |1\rangle \rightarrow 33.33333\%, |2\rangle \rightarrow 33.33333\%$

Starting Stage 4

Applying Shift Gate to Qutrit 4 with Qutrit 1 as control

$(0.5773502691896258 + i0.0, -0.28867513459481275 + i-0.5000000000000001, -$



$0.28867513459481314 + i0.4999999999999999994$ )

$|0\rangle \rightarrow 33.33333\%$ ,  $|1\rangle \rightarrow 33.33333\%$ ,  $|2\rangle \rightarrow 33.33333\%$

Applying Shift Gate to Qutrit 5 with Qutrit 2 as control

$(0.5773502691896258 + i0.0, -0.28867513459481314 + i0.4999999999999999994, -$

$0.28867513459481275 + i-0.5000000000000000001)$

$|0\rangle \rightarrow 33.33333\%$ ,  $|1\rangle \rightarrow 33.33333\%$ ,  $|2\rangle \rightarrow 33.33333\%$

Starting Stage 5

Applying Shift Gate to Qutrit 5 with Qutrit 1 as control

$(0.5773502691896258 + i0.0, -0.28867513459481314 + i0.4999999999999999994, -$

$0.28867513459481275 + i-0.5000000000000000001)$

$|0\rangle \rightarrow 33.33333\%$ ,  $|1\rangle \rightarrow 33.33333\%$ ,  $|2\rangle \rightarrow 33.33333\%$