Quantum Computing

Fryad Muhammad Rashid
School of Computing, Clemson University, Clemson, USA
fryadm@g.clemson.edu

ABSTRACT

This paper presents basic understanding of quantum computing, which is one of the hot topics in today's research area. The discussion starts to clarify the basic difference between a quantum computer and a classical computer at different levels such as bit level, register level, logic gate level, and circuit level. The main purpose of this paper is to deepen the understanding of the basic model of quantum computing by building a relationship between quantum gates, reversible logic gates, and classical logic gates. This paper demonstrates the quantum bit or qubit, quantum register, and quantum gates because they are important elements in the understanding of the quantum circuit model. The quantum computer simulator has been used to implement quantum circuits and quantum algorithms. The famous Deutsch's algorithm has been described and a corresponding quantum circuit is presented. The last demonstration is converting a combinational logic circuit to a quantum circuit for the specific scenario.

Keywords: Quantum circuits, Reversible logic gates, Classical logic gates, Algorithms in quantum computing.

INTRODUCTION

Nowadays, quantum computing is one of the big challenges in computer science and engineering because it consists of intersecting computer science and quantum physics. As we know classical computers are currently used to perform the computations that we require and they do it very efficiently. However, there are a set of problems which can not be calculated by classical digital computers in reasonable time. For that reason, researchers are trying to develop quantum computers by using a mechanical phenomenon to operate on the data and solve a particular problem. It is exponentially faster than classical computers and can be used to solve different types of problems in different fields very efficiently. The main purpose of quantum computation does not implement on accelerating digital computations by using quantum effects. The aim is to use quantum algorithms to solve these problems that can not otherwise be implemented on digital computers due to the fact that there are a lot of problems in computer science that are hard to control or execute on computers. For that purpose, quantum computing is applied in DNA computing, nanotechnology, trapped ions, superconductors, and photonics [3,10].

METHODOLOGY

This section describes basic concepts in quantum computing.

Qubits

As we know, the fundamental block of information for the classical computer is a bit. It can be represented by 0 or 1. At the same time, the basic building block of information for the quantum computer is the qubit or quantum bit. In this case, we have to clarify the basic difference between a classical bit and qubit. The classical bit represents voltage value which can be either low or high for the specific time, while qubit represents an electronic state
of the atom. As a result, apart from being in two states, any electron can be in superposition in two states, it can be in both 0 and 1 states simultaneously. Finally, the qubit can be represented in the superposition of its two logical states 0 and 1. This is called linear combination of two states: \(|\Psi\rangle = \alpha |0\rangle + \beta |1\rangle\), where \(\alpha\) and \(\beta\) are amplitude or complex numbers. This equation is useful in measuring the state of the qubit. We will find 0 with probability \(|\alpha|^2\) and 1 with probability \(|\beta|^2\). Thus \(|\alpha|^2 + |\beta|^2 = 1\). This is called normalization [2,1].

Quantum Register

It represents multiple qubits. In classical computers it is very easy to represent a register that contain two bit. These two bits have four different values to store at a given point of specific time. In a quantum computer, two qubits can be represented in the quantum register to build superposition for all of the four quantum states at a given point of specific time. It is represented as follows: \(|\Psi\rangle = \alpha_{00} |00\rangle + \alpha_{01} |01\rangle + \alpha_{10} |10\rangle + \alpha_{11} |11\rangle\) [10,2].

Direct Notation

It is used to represent qubits. As we know in quantum computing, 0 state is the ground state. It can be represented as \(|0\rangle\) and then 1 state is the excited state represented as \(|1\rangle\). The quantum register for two qubits can be represented in four states which are \(|00\rangle, |01\rangle, |10\rangle\) or \(|11\rangle\). The symbol \(|\rangle\) (called ket notation) is used to represent computational basis state [3,1].

QUANTUM GATES

Quantum logic gates are operators working on qubits and quantum registers. They change their states by rotating their corresponding state vectors. It means, they work on superposition of states. Quantum logic gates act on qubits such as one qubit, two qubits, and three qubits. In addition, the output of the quantum gate can be obtained by multiplying the operator matrix of the gate with the row vector of the input. Also, one of the differences between a classical logic gate and a quantum logic gate is that the quantum gate is reversible if we know the output we can identify the input. It is not true for classical gates except NOT gate. In this way, we can measure the result and we can reverse the computational process to find the input[3]. The following gates are very common in quantum computing [4-10]:

1. Hadamard Gate: It can be represented by H, acts on one qubit. The circuit symbol and the action of the H gate are presented below.

![Fig.1 Circuit symbol Hadamard gate](https://ssrn.com/abstract=3569594)

| \(|q_1\rangle\) | \(|q_0\rangle\) |
|----------------|----------------|
| \(|0\rangle\) | \(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\) |
| \(|1\rangle\) | \(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\) |
| \(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\) | \(|0\rangle\) |
| \(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\) | \(|1\rangle\) |

As shown in Fig.1, \(|q_1\rangle\) represents the state of the qubit before action of the gate and \(|q_0\rangle\) represents the new state after action of the gate as shown in table 1. It means, they are input and output as classical logic gates. The probability measure for both base states are equal to 0.5 in the state superposition.

2. Controlled- NOT (CNOT) Gate: It acts on two-qubits quantum register.
As shown in Fig.2, $|c\rangle$ is the upper qubit, it represents control qubit and $|t\rangle$ is the lower qubit and it represents the target qubit. The CNOT gate acts on the target qubit and reverses its state only if the control qubit is in state $|1\rangle$. Also, the state of the $|c,t\rangle$ represents a quantum register before action and the state of the $|c,o,t\rangle$ represents a quantum register after action respectively as shown in table2a.

![CNOT gate](image1)

**TABLE2a Action table of CNOT gate**

| $c,t$ | $c,o,t$ |
|-------|--------|
| $|00\rangle$ | $|00\rangle$ |
| $|01\rangle$ | $|01\rangle$ |
| $|10\rangle$ | $|11\rangle$ |
| $|11\rangle$ | $|10\rangle$ |

Through reversible logic gate perspective CNOT gate is called "Feynman" gate. It is a 2x2 quantum logic gate as shown in Fig.2 (b, c)[4].

![Feynman gate](image2)

The input vector for Feynman gate is $I(A, B)$ and the output vector is the $Q(P, Q)$. At the same time, output is defined by $P=A$, and $Q=A \oplus B$. In addition, the quantum cost is 1.

![Feynman gate circuit](image3)

This gate is helpful to duplicate the output. The truth table and classical logic circuit for Feynman gate are shown in Fig.2d.

![Feynman Classical logic circuit](image4)

**Table2b Action table of Feynman gate**

| A | B | P | Q |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 |

3. **Controlled-Controlled- NOT (CCNOT) Gate:**

It acts on a three-qubits quantum register. As shown in Fig.3, $|c_2\rangle$ and $|c_1\rangle$ are two upper control qubits and $|t\rangle$ is the lower target qubit. The CCNOT gate acts on target qubit and reverses its state only if both control qubits are
in state $\ket{I}$. Also, the state of the $\ket{c_{2i} c_{1i} t_{1i}}$ represents quantum register before action and the state of the $\ket{c_{2o} c_{1o} t_{0}}$ represents quantum register after action respectively as shown in table 3a.

Also, Tofolli gate input vector is $I(A, B, C)$ and the output vector is the $Q(P, Q, R)$. At the same time, output is defined by $P=A$, $Q= B$, and $Q= AB \ XOR \ C$ [8]. In addition, the quantum cost of F gate is 5. The truth table and classical logic circuit for Tofolli gate are shown below.

### Table 3a: Action table of CCNOT gate

| $c_{2i}$ $c_{1i} t_{1i}$ | $c_{2o} c_{1o} t_{0}$ |
|-------------------------|------------------|
| $\ket{000}$             | $\ket{000}$      |
| $\ket{001}$             | $\ket{001}$      |
| $\ket{010}$             | $\ket{010}$      |
| $\ket{011}$             | $\ket{011}$      |
| $\ket{100}$             | $\ket{100}$      |
| $\ket{101}$             | $\ket{101}$      |
| $\ket{110}$             | $\ket{110}$      |

Through reversible logic gate perspective CCNOT gate is called " Tofolli " gate. It is a 3x3 quantum logic gate as shown in Fig.3 (b, c) [4].
upper control qubit, and $|t_2\rangle$ and $|t_1\rangle$ represent the lower target qubits. The Fredkin gate swaps states of both target qubits if the state of control qubits is $|I\rangle$. Also, the state of the $|c_i t_{2i} t_{1i}\rangle$ represent the quantum register before action and the state of the $|c_o t_{2o} t_{1o}\rangle$ represent the quantum register after action respectively as shown in table 4a.

![Circuit symbol F gate](#)

**Fig.4a Circuit symbol F gate**

| Table 4a: Action table of F gate |
|----------------------------------|
| $|c_i t_{2i} t_{1i}\rangle$ | $|c_o t_{2o} t_{1o}\rangle$ |
| 000 | 000 |
| 001 | 001 |
| 010 | 010 |
| 011 | 011 |
| 100 | 100 |
| 101 | 110 |
| 110 | 111 |
| 111 | 111 |

Through a reversible logic gate perspective, Fredkin gate has a 3x3 quantum logic gate as shown in Fig.4 (b, c).

![Block diagram F gate](#)

**Fig.4b Block diagram F gate**

![F gate circuit](#)

**Fig.4c F gate circuit**

The truth table and classical logic circuit for F gate are shown below.

![F logical circuit](#)

**Fig.4d F logical circuit**

| Table 4b: Action table of F gate |
|----------------------------------|
| A | B | C | P | Q | R |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |
QUANTUM COMPUTING

The basic understanding of the circuit model in quantum computation is that the input is the initial state of the quantum register. The computation procedure is applied on the quantum logic circuit gate by the gate as a cascade at the end of the final state of the quantum register. The computation result is measured. This procedure is called a computational procedure. It means in each step quantum gates act on a quantum register. One of the important steps is the action of the gates in each step combined by a tensor product of Hilbert space operators to represent a state of the quantum register before action and the new state of quantum register after the action [9].

This paper develops a quantum computer simulator (QCAD) to simulate quantum circuits on classical computers. In addition, the inputs to simulator are the number of qubits in the quantum register, the number of computational procedure or steps, the initial state of the qubits in the form of quantum register and quantum logic gates applied at each step. All these inputs are entered into the simulator [12]. The algorithm of the simulator runs these inputs as described below:

1. Start:
2. Read:
   A. Number of qubits.
   B. Computational procedure or steps.
   C. Initial state of quantum register.
   D. Gate configuration in the quantum circuit for each step.
3. Process:
   A. Calculate tensor product of the Hilbert space for the initial state of quantum register.
   B. Select 1st computation procedure \((j=1)\).
   C. Calculate tensor product of the Hilbert space for quantum gate matrices that applied at the \(j\)th computational procedure.
   D. Calculate new state of quantum register.
   E. Condition: if the number of computational procedure is less than or equal to the number defined by the developer go to step Process(C), else continue.
4. Print:
   A. Print out probability of amplitudes for each computational procedure.
   B. Print out final result of quantum register state.
   C. Generate the graph for the output.
5. End:

![Quantum Circuit Diagram](https://ssrn.com/abstract=3569594)

The quantum computation shown in Fig.5 consist of three qubits and five steps. The computational procedure for every step has been clarified as follows:

**Step one:** It is the initial state of quantum register, \(|111\rangle\).

**Step two:** H gate is applied to upper qubit (Q1) and no gate to lower qubits.

**Step three:** CCNOT gate is applied to (Q1 and Q2) as a qubit control and (Q3) as a target.

**Step four:** Same gate applied to fourth step.

**Step five:** H gate is applied to control qubit (Q1). These steps are shown in Fig.5 (a) upper part of the Fig.5 is (a) as a quantum circuit.
In the lower part of Fig. 5 (b) the matrix is shown. The column of this matrix presents a computational procedure and the row presents quantum register states, which are written in ket notation to represent the state and decimal number also. The mapping procedure from a quantum circuit into matrix states to measure probability for each state has been clarified as follows:

**Matrix Step one:** The probability measurement of the quantum register at *Step one* that gives us this state $|111\rangle$, is equal to 1. At the same time, all of the matrix element in the first column are zero except the last state which contains 1.

**Matrix Step two:** The H gate applied at *Step two*. As we know H gate probability measure is 0.5 for state 1 and state 0 based on basis-state superposition. Therefore, the measurement of the quantum register state at the end of this step will give states $|011\rangle$ and $|111\rangle$ with probability 0.5 for each one.

**Matrix Step three:** The CCNOT gate applied at *Step three*. As we know that the CCNOT gate probability measure depends upon control qubits (Q1 and Q2) to change the state of target qubit. In this case, we have two states that came from H gate which are $|011\rangle$ and $|111\rangle$. They will become an input to CCNOT gate. According to CCNOT state table rules, the state $|011\rangle$ remains the same because first bit is zero and does not make any change in the target qubit, but the state $|111\rangle$ will change to $|110\rangle$ because first and second control qubits are 1 which will make a change in target qubit from state 1 to state 0. In this case, the probability measure for $|011\rangle$ and $|110\rangle$ is 0.5 for each one.

**Matrix Step four:** Again CCNOT gate applied at *Step four*. In here, output of first CCNOT becomes input to second CCNOT. Now, we have two input states $|011\rangle$ and $|110\rangle$. According to CCNOT base state rules, the state $|011\rangle$ remains the same, but the state $|110\rangle$ will change to $|111\rangle$ due to the same reason that we described in *Matrix Step three*. In this case, the probability measure for $|011\rangle$ and $|111\rangle$ is 0.5 for each one.

**Matrix Step five:** Finally, H gate before result. Now, output of the second CCNOT becomes input to H gate. In here, the output result showed that the highest probability measure is the state $|111\rangle$ which is 1.

Finally, the graphical representation for this matrix is shown in Fig. 5 (c). The *x-axis* represents the computational procedure which is the column of the matrix, and the *y-axis* represents possible states of the quantum register which is row of the matrix, and *z-axis* represents probability for each quantum register states.

![Quantum Computing Result](https://ssrn.com/abstract=3569594)

**CONVERT LOGICAL CIRCUITS TO QUANTUM CIRCUITS:**

This part of the paper concentrates on conversion of a classical logic circuit into a
quantum circuit. For that purpose, researchers should know how to design a combinational logic circuit if you have a scenario that requires it.

**Case Study:** Our case study is designed such that a combinational logic circuit for the following scenario is implemented and then changed to a quantum circuit "A committee of three individuals decided on issues for the organization. Each individual votes either yes or no for each proposal that arises. A proposal is passed if it receives at least two yes votes. Design a circuit that determines whether a proposal passes."

The main idea to solve the above scenario is that the user should build a truth table for it, and then use a K-map minimization technique to simplify the circuit. Finally, we design the logic circuit for it.

**TABLE 5** Truth table for scenario.

| Individual 1 (X) | Individual 2 (Y) | Individual 3 (Z) | Proposal (F) |
|------------------|-------------------|------------------|--------------|
| 0                | 0                 | 0                | 0            |
| 0                | 0                 | 1                | 0            |
| 0                | 1                 | 0                | 1            |
| 0                | 1                 | 1                | 1            |
| 1                | 0                 | 0                | 0            |
| 1                | 0                 | 1                | 1            |
| 1                | 1                 | 0                | 1            |
| 1                | 1                 | 1                | 1            |

**TABLE 6** K-map for scenario

| X | YZ | 00 | 01 | 11 | 10 |
|---|----|----|----|----|----|
| 0 | 1  | 1  | 0  | 0  | 0  |
| 1 | 1  | 1  | 1  | 1  | 1  |

\[ F = XY + XZ + YZ \]

**TABLE 7** Converting table.

| Classical Logic Gates | Reversible Quantum Gates |
|-----------------------|--------------------------|
| **AND gate**          |                          |
| \( a \)               | \( b \)                  |
| \( 0 \)               | \( a \land b \)          |
| **OR gate**           |                          |
| \( a \uplus b \)      |                          |
| \( b \)               | \( a \uplus b \)         |
| \( 0 \uplus 1 \)      | \( a \uplus b \)         |

Finally, the designed logic circuit is converted into a quantum circuit as shown in Fig. 7.

**DEUTSCH'S QUANTUM ALGORITHM:**

Deutsch discovered that for most problems quantum computers can solve at a faster rate than classical computers, however not exponentially faster. It is a studied conflict to show how quantum computers can be used. The problem posed by Deutsch is that: “Determine whether a given Boolean function \( f(x) \) is a constant, i.e. \( f(0) = f(1) \), or balanced,
i.e. $f(0) \neq f(1)$. "[14]. For this classical algorithm it requires two evaluations of $f$. This type of algorithm uses only one quantum evaluation by computing $f(0)$ and $f(1)$ simultaneously. The quantum circuit of the Deutsch's algorithm is as shown below:

![Deutsch's algorithm quantum circuit](image8)

The procedures to analyze Deutsch's algorithm is described below:

1. Initialize with $|\Psi_0\rangle = |01\rangle$
2. Build superposition of the $x$ states by using the first Hadamard (H) gate and set $y$ as a control input using the second H gate.
3. Calculate $f(x)$ using the special unitary circuit $U_f$ such as CNOT.
4. Check the $|\Psi_2\rangle$ states using the third H gate.
5. Measure probability of the $x$ qubit.

Deutsch algorithm has been implemented by using a quantum computing simulator such as QCAD to see the result of the algorithm.

![Deutsch algorithm implementation](image9)

The measure probability of the two qubits are shown below in table8.

| TABLE8 Deutsch result |
|------------------------|
| Input states | Probability | Deutsch algorithm |
|----------------|-------------|-------------------|
| $|0\rangle$ | 0.000000 | Constant: $f(0) = f(1)$ |
| $|1\rangle$ | 1.000000 | Balanced: $f(0) \neq f(1)$ |

CONCLUSIONS AND FUTURE WORK

It can be concluded that equivalent gates are found in three different topics such as quantum gates, reversible gates and classical logic gates. For example, CNOT gate in quantum computing is equivalent to Feynman gate in reversible logic gates, and both are equivalent to XOR gate in classical logic gates. These various subjects give us another conclusion. Quantum computers cannot replace classical computers because they are not universally faster. They are only faster for special type of calculations if all quantum super positions are available. At the same time, researchers have to do some kind of computation of parallelism. For that reason, we should not consider quantum computers to be faster for all applications. In fact, every operation is going to be slower than in desk computers. Finally, replacing a classical logic circuit by a reversible circuit is very useful because reversible circuits use very lower power in the design, but replacing logical circuit by quantum circuits is not helpful because the circuit becomes bigger and slower in operations. In future work, we will implement another two famous quantum algorithms which are the Grover's algorithm used for searching unstructured databases and Shor's algorithm used for factoring large numbers, by using different types of quantum computing simulators. Some simulators use programming languages such as QuIDE using C# and quantum computing play ground simulator that are developed by Google using Java scripts to build quantum circuits and vise versa.
REFERENCES

[1] Chakraborty P., Parkerson J. and Lala P., "Adder Designs using Reversible Logic Gates," WSEAS transactions on circuits and systems. ISSN: 1109-2734, vol. 9, issue 6, June, 2010.

[2] Goyal P., "Quantum Computing," Indian Institute of Technology, Delhi, 2012.

[3] Chiwande S. and Yelekar P., "Introduction to Reversible Logic Gates and its Application," 2nd national conference on information and communication technology (NCICT ) and Proceeding published in international journal of computer applications (IJCA), 2011

[4] Kumar K., Kiran P., and Garipelly R., "A Review on Reversible Logic Gates and their Implementation," international journal of emerging technology and advanced engineering, vol 3, issue 3, pp.417-423, ISSN: 2250-2459, ISO:9001:2008 certified journal, March 2013.

[5] Singh J., Nayak Sit., and Nayak Sha.,"An Introduction to Basic Logic Gates for Quantum Computer," international journal of advanced research in computer science and software engineering, vol. 3, issue 10 , ISSN: 2277-128X, October 2013.

[6] Barila A., "From classical computing to quantum computing," 12th international conference on development and application systems, Suceava, Romania, 15-17 May, IEEE, 2014.

[7] Beth T., "Quantum Computing: An Introduction," ISCAS international symposium on circuits and systems, Geneva, Switzerland, 28-31 May, IEEE, 2000.

[8] Nagvekar S., Rane A., Deshpande M., Rao A., and Shah H., "Implementation of High Speed Low Power Combinational and Sequential Circuits using Reversible Logic," Vivekanand Education Society's Institute of Technology, Mumbai-400072, 2010.

[9] Ranganathan N., and Thapliyal H., "Design of Reversible Sequential Circuits Optimizing Quantum Cost, Delay, and Garbage Outputs," Vol. 6, No.4, Article 13, Journal of Emerging Technologies in computer systems, December, ACM ,2010

[10] Hagouel P. and Karafyllidis, "Quantum Computers: Registers, Gates, and Algorithms," Proc 28th international conference on microelectronics (MIEL 2012), Serbia, pp. 15-21, 13-16 May, IEEE, 2012.

[11] Mano M., and Ciletti M., "Digital Design", 4E, Prentice Hall, 2007

[12] Website QCAD simulator: http://qcad.osdn.jp/

[13] William C.," Explorations in Quantum Computing, " DOI 10.1007/978-1-84628-887-6_2, Springer-Verlag London Limited, 2011

[14] Hayes J., "Quantum Logic Circuits," Advanced Computer Architecture Laboratory, University of Michigan, EECS Department, October, 2001.