Operations Algorithms on Quantum Computer

Moayad A. Fahdil[†], Ali Foud Al-Azawi[††], and Sammer Said [†††]

[†] Information Technology Faculty, Philadelphia University, Amman, Jordan
[††] Information Technology Faculty, Philadelphia University, Amman, Jordan
[†††] Al-Mansour University College Baghdad-Iraq

Abstract
A quantum computer is a device for computation that makes direct use of distinctively quantum mechanical phenomena, such as superposition and entanglement, to perform operations on data. In a classical (or conventional) computer, the amount of data is measured by bits, in a quantum computer; it is measured by quantum of bits (qubits). This work describes, and implements the universal quantum logic gates, the terminology is introduced with two well known quantum gates, the quantum NOT, and the quantum XOR gates. The NOT and XOR gates have already been described in classical reversible logic.

This work gives the implementation of the quantum circuits, such as the Quantum Half-Adder Circuits, which consists of quantum control control not gate (CCNot) and quantum control not gate (CNot). Also, it describes and implements the Quantum Full-Adder Circuits, which consists of two Quantum Half-Adder Circuits and one control not gate (CNot).

Depending on the quantum circuits, one can implement the quantum basic arithmetic operations such as (addition, subtraction, multiplication and division). After each implementation, the computational complexity of each step is calculated.

Key words:
Quantum computing, Quantum Logic Gates, Truth Table, quantum mechanical, qubits, quantum mechanical, superposition.

1. Quantum Computer

A quantum computer is a device for computation that makes direct use of distinctively quantum mechanical phenomena, such as superposition and entanglement, to perform operations on data. In a conventional computer, the amount of data is measured by bits; in a quantum computer, it is measured by qubits. The basic principle of quantum computation is that the quantum properties of particles can be used to represent and structure data, and that devised quantum mechanisms can be used to perform operations with this data.

In quantum mechanics, the state of a physical system (such as an electron or a photon) is described by an element of a mathematical object called a Hilbert space.

A classical computer has a memory made up of bits, where each bit holds either a one or a zero. The device computes by manipulating those bits, i.e. by transporting these bits from memory to (possibly a suite of) logic gates and back. A quantum computer maintains a set of qubits.

A qubit can hold a one, or a zero, or a superposition of these. A quantum computer operates by manipulating those qubits, i.e. by transporting these bits from memory to (possibly a suite of) quantum logic gates and back.

A classical computer operates on a 3 bit register. At a given time, the state of the register is determined by a single string of 3 bits, such as "101". This is usually expressed by saying that the register contains a single string of 3 bits. A quantum computer, on the other hand, can be in a state which is a mixture of all the classically allowed states. The particular state is determined by 8 complex numbers. In quantum mechanics notation we would write:

$$|Q\rangle = a\,|000\rangle + b\,|001\rangle + c\,|010\rangle + d\,|011\rangle + e\,|100\rangle + f\,|101\rangle\, g\,|110\rangle + h\,|111\rangle$$

Where a, b, c, d, e, f, g, and h are complex.

A complex number $(\alpha+\beta i)$ is called (complex valued) *amplitude*, and each probability $(|\alpha|^2+|\beta|^2)$ is the absolute square of the amplitude, because it equals $|\alpha+ \beta i|^2$. The probabilities must sum to 1 [1, 2, 3].

2. Transformation Matrices [4, 9]

It is often the case in quantum computation that we know what the operation we want to perform is, in terms of its effect on the state vector of our system, but we don't know how to express the operation as a matrix. In this short note, we demonstrate how this can be done quite easily.

Let's consider a simple example: the truth table for the controlled-NOT gate is

0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Manuscript received January 5, 2010.
Manuscript revised January 20, 2010.

to express this as a matrix operator so that we can apply it easily to our qubit states. In the usual language, the CNOT gate transforms our basis vectors in the following way:

|00> → |00>; |01> → |01>; |10> → |11>; |11> → |10>

or, in vector form

$$\begin{pmatrix}1\\0\\0\\0\end{pmatrix}\mapsto\begin{pmatrix}1\\0\\0\\0\end{pmatrix};\quad\begin{pmatrix}0\\1\\0\\0\end{pmatrix}\mapsto\begin{pmatrix}0\\1\\0\\0\end{pmatrix};\quad\begin{pmatrix}0\\0\\1\\0\end{pmatrix}\mapsto\begin{pmatrix}0\\0\\0\\1\end{pmatrix};\quad\begin{pmatrix}0\\0\\0\\1\end{pmatrix}\mapsto\begin{pmatrix}0\\0\\1\\0\end{pmatrix}$$

The key to deducing the matrix which performs these transformations is to appreciate how we multiply matrices together. It is most convenient to express this in component notation, where write

$$b_i = \sum_j M_{ij} a_j .$$

The b_i is the components of the new vector, the a_i are the components of the original vector, and the M_{ij} are the elements of the matrix operator *M*. In this notation *i* label the rows of M and *j* labels the columns. Expanding the summation, yields

$$b_1 = M_{11}a_1 + M_{12}a_2 + M_{13}a_3 + \ldots .$$

Applying this to the example of the CNOT gate, gives

$$\begin{aligned} b_1 &= M_{11}a_1 + M_{12}a_2 + M_{13}a_3 + M_{14}a_4 \\ &= M_{11}a_1 \quad (\text{since all other } a_i \text{ are zero}) \end{aligned}$$

and since b1 = a1 = 1, we must have M11 = 1. Similarly, we find that M22 = M34 = M43 = 1, with all other Mij = 0. We write the matrix down as:

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

A useful way of looking at this is to label the rows and columns of the matrix with the basis states they correspond to. You simply label each row and column with the basis states in the usual order. The row labels label the basis states of the new vector; the column labels label the basis states of the old vector. So, for example:

$$|00\rangle \mapsto |11\rangle ; \quad |01\rangle \mapsto |00\rangle ; \quad |10\rangle \mapsto |01\rangle ; \quad |11\rangle \mapsto |10\rangle$$

can be translated into matrix form as follows:

	$\|00\rangle$	$\|01\rangle$	$\|10\rangle$	$\|11\rangle$
$\|00\rangle$	0	1	0	0
$\|01\rangle$	0	0	1	0
$\|10\rangle$	0	0	0	1
$\|11\rangle$	1	0	0	0

So we have that

$$M = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

and so

$$M\,|00\rangle = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

and so on. Other gates are equally easy. For the Hadamard gate:

$$|0\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) ; \quad |1\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) ,$$

We have

	$\|0\rangle$	$\|1\rangle$
$\|0\rangle$	1	1
$\|1\rangle$	1	-1

and for the more complex example of

$$\begin{aligned} |00\rangle &\mapsto \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\ |01\rangle &\mapsto \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \\ |10\rangle &\mapsto \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \\ |11\rangle &\mapsto \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \end{aligned}$$

We have

	$\|00\rangle$	$\|01\rangle$	$\|10\rangle$	$\|11\rangle$
$\|00\rangle$	1	0	0	1
$\|01\rangle$	0	1	1	0
$\|10\rangle$	0	1	-1	0
$\|11\rangle$	1	0	0	-1

3. Quantum Bits Representation

In this section, we describe how the bits are represented in quantum computer.

3.1. One-Quantum Bits [6, 7, 8, 9]

A qubit can exist in an arbitrary superposition state, a measurement on it will always find it in one of the two eigenstates, |0> or |1>, according to the measurement postulate of quantum mechanics.

$|Q> = a_0|0> + a_1|1>$

$$|a> = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix}$$

$$|0> = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

this is a superposition matrix for Dirac |0>

$$|1> = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

this is a superposition matrix for Dirac |1>

3.2. Two-Quantum Bits [6, 7, 8, 9]

If we have more than one qubit in our quantum system, we can express its state in terms of product eigenstates. For example, a two-qubit system has the basis states, |00>, |01>, |10>, |11>, which are the quantum analogs of the input lines in the truth table for a classical logic gate. Unlike classical bits however, two or more qubits can interfere with one another, creating a macroscopically coherent superposition, of the form

$c_{00}|00>$ + $c_{01}|01>$ + $c_{10}|10>$ + $c_{11}|11>$.

$|Q> = a_{00}|00> + a_{01}|01> + a_{10}|10> + a_{11}|11>$

$$|Q> = \begin{pmatrix} a_{00} \\ a_{01} \\ a_{10} \\ a_{11} \end{pmatrix}$$

$$|00> = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, |10\rangle$$

$$= \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, |11> = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

3.3. Representation of the QuBit

In this section, we will explain how Dirac representations convert to the superposition representation to deal it as quantum bits (QuBit) in quantum computers.

Below, we illustrate our algorithm for convert Dirac qubits representations to the superposition qubits representation.

Algorithm 1 convert

Input:

- qubit as a Dirac representation (Q_D).
- n= number of bits.

Output:

- qubit as a superposition matrix representation (Q_B).

Process:

Step1: set the superposition matrix (Q_B) by zero according to 2^n

Step2: let t= 1.
k= 1

Step3: Determine the position (t) of the superposition of the matrix.

Step4: compute this position (t) by the forloop as below

```
For i=n downto 1 do
Begin
      t= t+QD[i] *k;
      k=k*2;
 End;
```

Step5: set the value of position (t) by one in QB like $Q_B[t]$ =1.
Step6: Output the superposition matrix (Q_B).
Step7: Finish.

Below, we illustrate our algorithm for converting superposition qubits representations to the Dirac qubits representation.

Algorithm 2 deconvert

Input:
- qubit as a superposition matrix representation (Q_B).
- n= number of qubits

Output:
qubit as a Dirac representation (QD).

Process:

Step1: let t=0.
Step2: find the position (t) in the superposition of the matrix (Q_B) by the repeat until loop below.

```
Repeat
      t=t+1;
Until QD[t]=1;
```

Step3: compute the a Dirac matrix (Q_D) by the forloop below

```
For i=n downto 1 do
Begin
      QD[i] =t mod 2;
       t=t div 2;
 End;
```

Step4: Output the Dirac matrix (Q_D).
Step5: Finish.

4. Quantum Not and XOR Gates [8, 9, 10]

The problem of universality can be posed for quantum computation as well, in asking whether arbitrary unitary operations can be broken down into simpler ones. Similar to classical logic, quantum logic gates exist that operate on a handful of qubits at a time, and that are able to simulate arbitrary unitary operations. Note that it is a property of unitary matrices that a product of two of them remains unitary; hence a product of unitary logic gates will also be unitary. Before we describe the universal quantum logic gates, we introduce the terminology with two well known quantum gates, the NOT, and the XOR gates. We have already described the NOT and XOR gates in classical reversible logic. A straight-forward quantum generalization of these gates is the unitary matrices.

$$U_{NOT} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{matrix} |0\rangle \\ |1\rangle \end{matrix} \qquad U_{XOR} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{matrix} |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{matrix}$$

Which describes the evolution of the two product eigenstates of the one-qubit NOT gate, and the four products eigenstates of the two-qubit XOR gate. Note that unlike their classical counterparts, these quantum gates can transform superposition states as well. For example, operations of the form, U_{NOT}: $c_0|0> + c_1|1> \rightarrow c_0|1> + c_1|0>$, are also possible with the quantum NOT gate, we will describe the quantum Not Gate and quantum XOR Gate with details and algorithms in the next section.

4.1. One Input Quantum Bit Gate

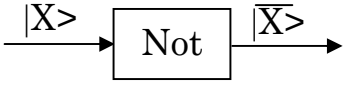
The behavior of quantum Not Gate is:-

$$\begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \rightarrow \begin{pmatrix} a_1 \\ a_0 \end{pmatrix}$$

But the unitary matrix of quantum Not Gate is:-

$$U_{not} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

This is the block diagram of quantum Not Gate is:-



Here, we multiply the unitary matrix of quantum Not Gate with |0> that represent with superposition $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ or with |1> that is represented with superposition $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ to get the output of this gate.

$$\therefore \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad \text{i.e } U_{not}\,|0> = |1>$$

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \text{i.e. } U_{not}\,|1\rangle = |0\rangle$$

Bellow, we illustrate our algorithm of quantum **Not gate**:-

Algorithm 3 Quantum Not Gate

Input:
qubit as a Dirac representation (Q_D).
Output:
quantum Not gate result.

$$U_{QNot} = \begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix}$$

Process:

Step1: set the unitary matrix transformation

Step2: represent the sequence of qubit as one dimension ($Q_B[i]$) by using algorithm 4.1 convert.

Step3: compute the quantum Not of qubit by multiplying

The U_{QNot} matrix by Q_B
$N_{QB} = Q_B * U_{QNot}$.

Step4: Output the (N_{QB}).
Step5: Finish.

4.2. Two Input Quantum Bit Gate

When we use two input quantum Bit in gate, this case is called controlled not (CNOT) gate, this process is equivalent to XOR gate in classical computer, below we present the unitary matrix of quantum XOR Gate or quantum controlled Not Gate (CNOT) with its coefficients.

$$U_{cnot} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} a_{00} \\ a_{01} \\ a_{10} \\ a_{11} \end{pmatrix} = \begin{pmatrix} a_{00} \\ a_{01} \\ a_{11} \\ a_{10} \end{pmatrix}$$

Here, we multiply the unitary matrix of quantum controlled Not

Gate with $|00\rangle$ is represented in superposition $\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$

Here, we multiply the unitary matrix of quantum Controlled Not

Gate with $|10\rangle$ that is represented in superposition $\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$

$$U_{cnot}\,|10\rangle = |11\rangle$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Here, we multiply the unitary matrix of quantum Controlled Not

Gate with $|11\rangle$ that is represented in superposition $\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$

$$U_{cnot}\,|11\rangle = |10\rangle$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

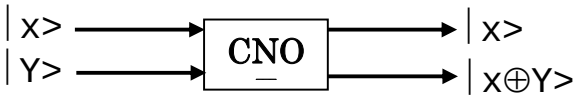
Here, we multiply the unitary matrix of quantum Controlled Not

Gate with $|01\rangle$ that is represented in superposition $\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$

$$U_{cnot} \mid 01> = \mid 01>$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

This is the block diagram of quantum Controlled Not Gate when it has the two inputs (|x> , |y>) and we can get from it two outputs (|x> , |x> ⊕ |y>).



Now, we illustrate our **algorithm** of quantum **Control Not gate**:-

Algorithm 4 Quantum Control Not Gate

Input:

qubit as a Dirac representation (Q_D).

Output:

quantum control Not gate result.

Process:

Step1: set the unitary matrix transformation

$$U_{CNot} = \begin{matrix} 1\ 0\ 0\ 0 \\ 0\ 1\ 0\ 0 \\ 0\ 0\ 0\ 1 \\ 0\ 0\ 1\ 0 \end{matrix}$$

Step2: represent the sequence of qubit as one Dimension(Q_B[i]) by using algorithm 4.1 convert.

Step3: compute the quantum control Not of qubit by multiplying

The U_{CNot} matrix by Q_B

$$CN_{QB} = Q_B * U_{CNot} .$$

Step4: Output the (CN_{QB}).

Step5: Finish.

4.3. Three Input Quantum Bits Gate

This section, represents and describes the three inputs of quantum bits, which are used in controlled CNOT (CCNOT), below we present the unitary matrix of quantum controlled CNot Gate with its coefficients. It is also called controlled controlled NOT (CCNOT) or called Toffoligate.

The superposition of this qubits, is represented in the next description:

Here, we multiply the unitary matrix of quantum Controlled CNot

Gate with | 000> that is represented in superposition $(1\ 0\ 0\ 0\ 0\ 0\ 0\ 0)^T$

$$U_{ccnot} \mid 000> = \mid 000>$$

Here, we multiply the unitary matrix of quantum Controlled CNot

Gate with | 000> that is represented in superposition $(0\ 1\ 0\ 0\ 0\ 0\ 0\ 0)^T$

$$U_{ccnot} \mid 001> = \mid 001>$$

Here, we multiply the unitary matrix of quantum Controlled CNot

Gate with | 000> that is represented in superposition $(0\ 0\ 1\ 0\ 0\ 0\ 0\ 0)^T$

$$U_{ccnot} \mid 010> = \mid 010>$$

Here, we multiply the unitary matrix of quantum Controlled CNot

Gate with | 000> that is represented in superposition $(0\ 0\ 0\ 1\ 0\ 0\ 0\ 0)^T$

$$U_{ccnot} \mid 011> = \mid 011>$$

Here, we multiply the unitary matrix of quantum Controlled CNot

Gate with | 000> that is represented in superposition $(0\ 0\ 0\ 0\ 1\ 0\ 0\ 0)^T$

$$U_{ccnot} \mid 100> = \mid 100>$$

Here, we multiply the unitary matrix of quantum Controlled CNot

Gate with $\mid 000>$ that is represented in superposition $(0\ 0\ 0\ 0\ 0\ 1\ 0\ 0)^T$

$$U_{ccnot} \mid 101> = \mid 101>$$

Here, we multiply the unitary matrix of quantum Controlled CNot
Gate with $\mid 000>$ that is represented in superposition $(0\ 0\ 0\ 0\ 0\ 0\ 1\ 0)^T$

$$U_{ccnot} \mid 110> = \mid 111>$$

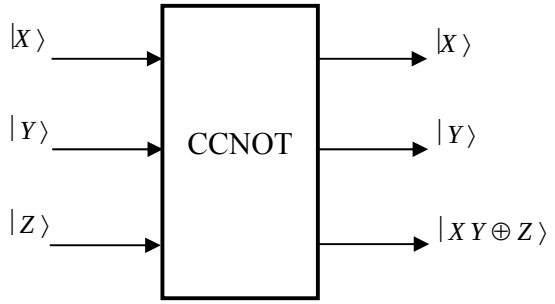
Here, we multiply the unitary matrix of quantum Controlled CNot

Gate with $\mid 000>$ that is represented in superposition $(0\ 0\ 0\ 0\ 0\ 0\ 0\ 1)^T$

$$U_{ccnot} \mid 111> = \mid 110>$$

Below, the block diagram of quantum CCNot Gate represents:

-Three inputs: **(** $\mid X\rangle, \mid Y\rangle, \mid Z\rangle$ **)**.
-Three outputs: **(** $\mid X\rangle,\ \mid Y\rangle,\ \mid X\,Y \oplus Z\rangle$ **)**.



Now, we illustrate our **algorithm** of quantum **Control Control Not gate**:-

Algorithm 5 Quantum Control Control Not Gate

Input:
qubit as a Dirac representation (Q_D).

Output:
quantum control Not gate result.

Process:

Step1: set the unitary matrix transformation

Step2: represent the sequence of qubit as one dimension($Q_B[i]$) by using algorithm 4.1 convert.

Step3: compute the quantum control controNot of qubit by

Multiplying The U_{CCNot} matrix by Q_B
$CCN_{QB} = Q_B * U_{CCNot}$.

Step4: Output the (CCN_{QB}).
Step5: Finish.

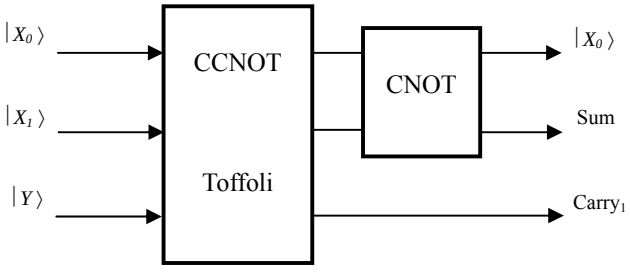
$$U_{CCNot} = \begin{matrix} 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0 \\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0 \\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0 \\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0 \\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0 \end{matrix}$$

5. Quantum Circuits Gate Representation

In this section, we describe the quantum circuits gate that is represented by Half Adder quantum circuit and Full Adder quantum circuit.

5.1. Half Adder Quantum Circuits

We design the Quantum Half-Adder Circuits, which consists of quantum control control not gate (CCNot) and quantum control not gate (CNot), below is the structure design of Quantum Half-Adder Circuits with its table.



This Circuit represents the three inputs ($\mid X_0\rangle,\ \mid X_1\rangle,\ \mid Y\rangle$) and three outputs ($\mid X_0\rangle$, Sum, $Carry_1$) .

The result of the Half-Adder:-
Sum $= |X_0\rangle \oplus |X_1\rangle$
Carry$_1$ = Carry $\oplus |Y\rangle$
$= |X_0 X_1 \oplus Y\rangle$
Let $|Y\rangle$ = zero
Carry$_1$ $= |X_0 X_1 \oplus 0\rangle$

See the table below

X_0	X_1	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Below, we illustrate our **algorithm** of **Half Adder Quantum Circuits**:-

Algorithm 6 Quantum Half-Adder Circuit

Input:
Two qubit as a Dirac representation ($|X_0>$, $|X_1>$).
Output:
Two qubit result (Sum,Carry).
Process:
Step1: let $|Y\rangle = |0\rangle$

Step2: compute the output of Quantum Control Control Not by using algorithm 4.5 to get the carry carry = $|X_0X_1 \oplus Y>$

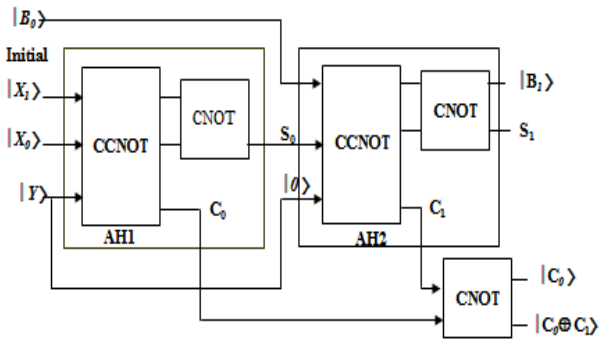
Step3: compute the output of Quantum Control Not by using algorithm 4.4 to get the sum
Sum = $|X_0\rangle \oplus |X_1\rangle$

Step4: output the result of Quantum Half-Adder represented by
Sum and Carry.

Step5: Finish.

5.2. Full Adder Quantum Circuits

We design the Quantum Full-Adder Circuits, which consists of two Quantum Half-Adder Circuits and one control not gate (CNot), below is the structure design of Quantum Full-Adder Circuits.



The result of the Half-Adder one (AH1):-

Sum (S_0) = $|X_0\rangle \oplus |X_1\rangle$
Carry (C_0) = $|X_0 X_1 \oplus Y\rangle$
Let $|Y\rangle$ = zero
$\therefore$ Carry = $|X_0 X_1 \oplus 0\rangle$

The result of the Half-Adder two (AH2):-

Sum (S_1) = $|B_0\rangle \oplus |S_0\rangle$
Carry (C_1) = $|B_0 S_0 \oplus Y\rangle$
Let $|Y\rangle$ = zero
$\therefore$ Carry = $|B_0 S_0 \oplus 0\rangle$

The result of the full-Adder:-

Sum (S_1) = $|B_0\rangle \oplus |S_0\rangle$
Carry = $|C_0 \oplus C_1\rangle$

Below, we illustrate our **algorithm** of **Full Adder Quantum Circuits**:-

Algorithm 7 Quantum Full-Adder Circuit

Input:
Two qubit as a Dirac representation ($|X_0>$, $|X_1>$).

Output:
Two qubit result (Sum,Carry).

Process:
Step1: let $|Y\rangle = |0\rangle$

Step2: compute the output of Quantum Half-Adder1 (HA1) by using algorithm 4.6 to get

Sum (S_0) = $|X_0\rangle \oplus |X_1\rangle$
carry (C_0) = $|X_0X_1 \oplus Y>$

Step3: compute the output of Quantum Half-Adder2 (HA2) by using algorithm 4.6 to get

Sum (S_1) = $|B_0\rangle \oplus |S_0\rangle$
carry (C_1) = $|B_0 S_0 \oplus Y >$

Step4: compute the output of Quantum Control Not by using algorithm 4.4 to get

Sum = S1
Carry = $|C_0 \oplus C_1\rangle$

Step5: output the result of Quantum Full-Adder represented by

Sum and Carry.

Step6: Finish.

6. Quantum Arithmetic Operations on Quantum Computer

This section describes and represents how we can design and implement the basic arithmetic operations to quantum basic arithmetic operations to be suitable for use in the quantum computer, these operations are (Addition, Subtraction, multiplication and Division).

6.1. Quantum Addition Operation

The quantum addition operation can be performed by converting each number into sequence of qubit then adding these two sequences by using our algorithm of Quantum **Addition Operation**

Algorithm 8 Quantum Addition Operation

Input:
Two decimal numbers

Output:
The result of Addition as a decimal number.

Process:
Step1: Convert the first decimal number into dirac representation
$A= |a_1, a_2 , \ldots\ldots, a_n\rangle$

Step2: Convert the second decimal number into Dirac Representation

$B= |b_1, b_2 , \ldots\ldots, b_n\rangle$

Step3: compute the addition process by Quantum Full-Adder By using algorithm 4.7.

Step4: convert the dirac result of addition into decimal number.

Step5: Output the result.
Step6: Finish.

6.2. Quantum Subtraction Operation

The quantum subtraction operation can be performed by converting each number into sequence of qubit. Then take 2's complement of the second number. This process is performed by using our algorithm of Quantum **Subtraction Operation**

Algorithm 9 Quantum Subtraction Operation

Input:
Two decimal numbers

Output:
The result of subtraction as a decimal number.

Process :
Step1: Convert the first decimal number into dirac representation

$A= |a_1, a_2 , \ldots\ldots, a_n\rangle$

Step2: Convert the second decimal number into dirac representation

$B= |b_1, b_2 , \ldots\ldots, b_n\rangle$

Step3: convert the second dirac number into 2'S complement representation

B = B + 1

Step4: compute the addition process by Quantum Full-Adder By using algorithm 4.7.

Step5: convert the dirac result of addition into decimal number.

Step6: Output the result.
Step7: Finish.

6.3. Quantum Multiplication Operation

The multiplication operation can be performed by converting each number into sequence of qubit then multiplying these two sequences by using our algorithm of Quantum **Multiplication Operation**

Algorithm 10 Quantum Multiplication

Input:

Two decimal numbers

Output:

The result of multiplication as a decimal number.

Process :

Step1: Convert the first decimal number into dirac representation

$$A= |a_1,, a_2 \ , \ldots\ldots , a_n\rangle$$

Step2: Convert the second decimal number into dirac representation

$$B= |b_1,, b_2 \ , \ldots\ldots , b_n\rangle$$

Step3: set C = $|0\rangle$

Step4: compute the multiplication process by Following:

Step4.1: if dirac digit = $|1\rangle$ then compute the addition process by Quantum Addition By using algorithm 4.8.

Step4.2: Shift A to the left by zero according to case of Dirac of B.

Step4.3: goto to the Step4.1.

Step5: convert the dirac result of Multiplication into decimal number.

Step6: Output the result.

Step7: Finish.

6.4. Quantum Division Operation

The division operation can be performed by converting each number into sequence of qubit then dividing these two sequences by using our algorithm of Quantum **Division Operation**

Algorithm 11 Quantum Division Operation

Input:

Two decimal numbers

Output:

The result of division as a decimal number without reminder.

Process:

Step1: Convert the first decimal number into dirac representation

$$A= |a_1,, a_2 \ , \ldots\ldots , a_n\rangle$$

Step2: Convert the second decimal number into dirac representation

$$B= |b_1,, b_2 \ , \ldots\ldots , b_n\rangle$$

Step3: set C = $|0\rangle$

Step4: compute the division process by following

Setp4.1: do while A > = B

Step4.2: compute the subtraction process by Quantum subtraction By using algorithm 4.9.

Step4.3: increment C by one: C=C+1

Step5: convert the dirac result of Division into decimal number.

Step6: Output the result.

Step7: Finish.

7. Conclusions

This work introduces the quantum computer and computation principles and properties. It gives the implementation of the quantum circuits, such as the Quantum Half-Adder Circuits, which consists of quantum control control not gate (CCNot) and quantum control not gate (CNot). Also, it describes and implements the Quantum Full-Adder Circuits, which consists of two Quantum Half-Adder Circuits and one control not gate (CNot). The steps of implementation of quantum basic operations (addition, subtraction, multiplication division) with its algorithms are an essential step toward building the hardware and software of quantum computer.

Reference

[1] Wikipedia, the free encyclopedia, *Quantum Computer*, Center For Quantum Computation (CQC), 2005.

[2] Jacob West, *The Quantum Computer*, Published in Scientific of America, 2000.

[3] Eleanor Rieffel & Wolfgarg Polak, *An Introduction to Quantum Computer for Non-Physicists*, www.rieffelpal.xerox.com& & www.Polak.pal.xerox.com, 1998.

[4] John Preskill, *Reliable Quantum Computers*, California Institute of Technology,1997.

[5] Mohammad Inayatullah Babari, Shakell Ahmad, Sheeraz Ahmed, Iftikhar Ahmed Khani, and Bashir Ahmad, *Implementing Dimensional-View of 4X4 Logic Gate/Circuit for Quantum Computer Hardware using Xylinx*, JSSST Vol. 9 No. 5, December 2008.
[6] A.D. Manzano & L. Steinberg, *Idea of Quantum Computation*, Center for Quantum Computation(CQC), 1999.
[7] David Deutsch & Artur Ekert, *Machines, Logic and Quantum Physics*, Center for Quantum Computation (CQC), University of Oxford, 1999.
[8] Xinlan Zhou, Dabbie W. Leung & Isaac L. Chuang, *Methodology for quantum logic gate construction*, Stanford university, IBM Research and Stanford university, 2000.
[9] Ashok Muthukrishnan, *Classical and Quantum Logic Gates*, Rochester Center for Quantum Information (RCQI), 1999.
[10] Vipin Mishra, *Developing Innovative Programs for Quantum Computers and Algorithms for Interpretation of Complex Calculations*, Amity University, Amity Institute of Nanotechnology, Project Guide, 2006.