A REPORT

ON

A NOVEL 12-QUBIT MULTIPLE QUANTUM ERROR CORRECTING CODE

BY

Harsha Machineni	1700315C204
Vibhishan Ranga	1700365C204

prepared in the partial fulfilment of

PRACTICE SCHOOL III

AT

BML MUNJAL UNIVERSITY, GURGAON

A Practice School III Station of

BML MUNJAL UNIVERSITY
SCHOOL OF ENGINEERING AND TECHNOLOGY
17th MAY 2021

CERTIFICATE

This is to certify that Practice School Project of Harsha Machineni & Vibhishan Ranga titled " A Novel 12-Qubit Multiple Quantum Error Correcting Code " is an original work and that this work has not been submitted anywhere in any form. Indebtedness to other works/publications has been duly acknowledged at relevant places. The project work was carried during 11th January 2021 to 14th May 2020 at BML Munjal University.

Signature of Mentor	**Signature of HOD**
Name: Prof. Mohit Sinha	**Name:** Prof. Ajay Mohan Goel
Designation: Adjunct Professor, BMU	**Designation:** Head of Department (ECE), BMU
(Seal of the organization with Date)	(Seal of the organization with Date)

BML MUNJAL UNIVERSITY
PRACTICE SCHOOL – III
JOINING REPORT

Date:

Name of the Student	
Name and Address of the Practice School – III Station	
Department Allocated	
Name and Designation of the Industry Guide/ Industry Mentor for the Project	
Industry Mentor Contact No.	
Industry Mentor E-mail Address	

ACKNOWLEDGEMENT

The opportunity we had with Practice School III was a great chance for learning and professional development. Therefore, we consider ourselves as very lucky individuals to be provided with the opportunity and be a part of it.

We would like to use this opportunity to express our deepest gratitude and special thanks to our mentor and industry guide Prof. Mohit Sinha who in spite of being extremely busy with his duties, took time out to hear, guide and keep us on the correct path and allowing us to carry out our research at the required pace.

We express our deepest thanks to all faculty members for taking part in useful decisions and giving necessary advices and guidance and arranged all facilities to make this project easier. We choose this moment to acknowledge their contribution gracefully. It is also our sentiment to place on the record our best regards, deepest sense of gratitude to Dr. Ashok Suhag for his careful and precious guidance which was extremely valuable right at the inception of this initiative.

We consider this opportunity as a big milestone in our career development. We will strive to use gained skills and knowledge in the best possible way, and we will continue to work on their development, in order to achieve desired career objectives.

Also, thanks to the respective professors for their constant effort, understanding and making this entire course possible -

1) Dr. Manoj Arora (Vice Chancellor, BMU)
2) Dr. Maneek Kumar (Dean SOET)
3) Prof. Ajay Mohan Goel (HOD, Electronics and Communication Engineering)

Sincerely,

Harsha Machineni and Vibhishan Ranga

BML Munjal University, Gurugram

ABSTRACT

Quantum error correction (QEC) is a strict combination of Quantum Mechanics and Classical Error Correction (CEC). Principles from these branches of study are consolidated to form a study of error correction for quantum computation and quantum information theory. A classical Shor's 9-qubit error correcting code approach is extended to a novel 12-qubit multiple error correcting code being constructed for increased efficiency while preserving the authenticity of the information being transmitted.

Keywords Quantum Error Correction, Classical Error Correction, Quantum Gates, Pauli Gates, Shor's Algorithm, IBM Quantum Computer

I. Table of Contents

II. List of figures

III. List of tables

Introduction

In the classical world, events generally occur in a continuous fashion i.e. they progress in a smooth and orderly manner. Most importantly, they follow a predictable pattern. One may argue that when a coin is tossed, the probability of getting a head or tail (outcome) is exactly the same i.e. ½. But if we were to apply physical factors such as air resistance, gravity etc to this situation, we could predict the outcome of the toss. However, in the quantum world, events are totally unpredictable. These unpredictable transitions of quantum states are referred to as quantum leaps. These quantum leaps are seemingly discontinuous. The one theory that connects classical and quantum realms is the "Theory of Decoherence". This theory states that it is due to decoherence that qubits are extremely fragile and hence, the ability of them to stay entangled is endangered. This means for a quantum computer to exist, decoherence must be completely eliminated [1].

Quantum computation is a very important field of science with a great application of quantum information. Quantum computation can be employed in many areas and it holds a lot of practical applications. In order for these applications to be error free, which arise due to decoherence, quantum error correction (QEC) becomes a vital field of study. A detailed study of principles in classical error correction will enable us to use homogenous yet strict changes to quantum mechanics to construct efficient models for quantum error correction.

Today, error correcting codes are used frequently in communication systems and data storage systems since Shannon first demonstrated the mathematics in 1948 [9]. A model of such a system is shown in Figure 1. The source transmits the data through a channel. As the channel is imperfect, noise is added to the data being transmitted through the channel. This addition of noise leads to corruption of the data. To make information immune to such noises, advanced methods are used to protect the integrity of data. Redundant information is added at the time of transmission and when the receiver receives the data, it checks for these errors, corrects the errors and removes the redundant information. This strategy is referred to as error correcting codes (ECC) [2].

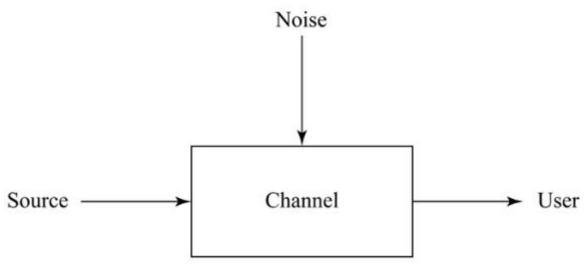


Figure 1 : Communication channel with noise/interference

1.1 CLASSICAL ERROR CORRECTION

The unit of information in modern computing devices is known as the bit, A bit may take the values 0 or 1. The most common errors associated with bits are the bit flip errors. This means, 0/1 would flip to 1/0 [1] [2].

The Repetition Code : This method involves repeating the bit many times. It is one of the most significant yet classical methods used. Here, the information bit is triplicated i.e. the repetition is as follows :

$$\begin{aligned} 0 &\rightarrow 000 \\ 1 &\rightarrow 111 \end{aligned} \tag{1}$$

Now, the received codewords/bits can be decoded by the principle of majority vote i.e. the bit is 0 if majority of the codeword is 0 and majority vote i.e. the bit is 1 if majority of the codeword is 1 [2]. If we received the set {001, 010, 100, 110, 101, 011}, the logic tells us the first three codewords would reduce to 0 as the majority bit is 0 and the last three codewords would return 1 as the majority bit is 1. This method allows reduction of errors while protecting the integrity of the information. The reason we do not consider two bit flips is the probability of encountering such a situation is almost zero i.e. the codeword {000} changing to say {110} is highly unlikely in the real world.

Linear codes : The repetition code encodes a bit into 3 bit codewords as shown in equation (1). This indirectly means we may use the matrix notations to represent these codewords as follows,

$$c_i = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} a_i \,,$$

Here, $a_i = [i]$ and $c_i = [i, i, i]^T$ and i takes the value 0 or 1. Also, $[i, i, i]^T$ is called the generator matrix for the repetition code. This matrix depicts the rule of encoding bits to codewords. By definition, a Linear code encoding k-bit messages into an m-bit code space is specified by an m by k generator matrix G whose entries are all elements of Z_2, that is, 0 or 1.

1.2 QUANTUM ERROR CORRECTION

Classical coding theory has ever been developing and has reached a very mature position where researchers were unable to figure out how to implement classical principles to quantum computation. In 1995-96, Peter Shor [6] and Steane [7] showed that QEC existed.

As written before, the methods applied to classical error correction are implored again with strict changes made which are analogous to the quantum domain. But, there is a challenge while doing this i.e. a similar logic to that of repetition code cannot be applied as quantum bits or qubits cannot be replicated due to the “No Cloning Theorem” which states the quantum states cannot be cloned [3]. Nielson et al. have mentioned the following challenges to quantum error correction :

- Measurement would destroy quantum information
- No cloning
- Errors are continuous as quantum states are continuous

These three points pose very different challenges when compared to classical error correction. This indicates that there is the requirement of defining new logical methods to solve the problem. Here, instead of using the logical gates in binary systems, new quantum gates devised by pioneers of quantum mechanics are implemented. The further section discusses these gates in great detail.

1.3 QUANTUM BITS

The unit of information in quantum computing is known as quantum bit or qubit [5]. A qubit behaves exactly like the classical bit enhanced by the superposition principle. Physical states are represented by a 'ray' in Hilbert Space H. These vectors are called 'kets'. The Hilbert Space is a normal vector space in which the concept of inner product is defined.

From a data processing perspective, a qubit's state may have two logic states i.e. kets, $|0\rangle$ and $|1\rangle$. Their Hermitian conjugates are denoted by 'bra' i.e. $\langle 0|$ and $\langle 1|$. The notation for these states was given by Dirac; known as the bra-ket notation [2]. Now, the inner product may be defined as "$\langle \ | \ \rangle$". Superposition of these states can be expressed as $\alpha|0\rangle + \beta|1\rangle$. Here, α and β are amplitudes of the superposition. The other way of representing the superposition is in the vector format as expressed in (2). Bennett and Shor [8] have elaborately demonstrated the difference between bits and qubits.

$$\alpha|0\rangle + \beta|1\rangle = \begin{bmatrix}\alpha \\ \beta\end{bmatrix} \tag{2}$$

1.4 QUANTUM GATES

Analogue to the way a classical computer is built using electrical circuits consisting of wires and primarily logic gates, a quantum computer is built the same way using quantum circuits consisting of wires and quantum gates to process quantum information or quantum data. Quantum gates vary on the basis of the number of inputs. Broadly, quantum gates are classified into single or multiple qubit gates.

Single qubit gates

X Gate [4] or the quantum NOT gate performs the function analogous to the classical NOT gate. However, the quantum NOT gate acts linearly, that is, it takes the state (3) to the state where the $|0\rangle$ and $|1\rangle$ have been interchanged as in (4),

$$\alpha|0\rangle + \beta|1\rangle, \tag{3}$$

$$\alpha|1\rangle + \beta|0\rangle \tag{4}$$

This linear behaviour is the general property of quantum mechanics. Mathematically,

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

$$\alpha|0\rangle + \beta|1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix},$$

$$X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} \tag{5}$$

The X Gate is also popularly known as Pauli-X Gate.

Z Gate [4] also known as the Phase gate is a selective gate. This gate leaves $|0\rangle$ unchanged and changes the phase i.e. flips $|1\rangle$ to $-|1\rangle$. The vector form of the gate is,

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{6}$$

The Z Gate is also popularly known as Pauli-Z Gate.

Hadamard Gate [4] is a special gate. It rotates the state by 90 degrees and reflects them about x-y plane. Mathematically,

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{7}$$

Here, the states transform as expressed in (8). It is also noteworthy that these transformations demonstrate qubits as they would be represented in the diagonal basis. The representation for the same are also expressed in (8).

$$|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \Rightarrow |+\rangle$$

$$|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \Rightarrow |-\rangle \tag{8}$$

Apart from the above single qubit quantum gates, there are few other gates which are vital in the study of quantum information theory and quantum error correction. Vector representation of these gates are as follows,

T Gate, also known as the $\pi/8$ gate is as follows,

$$\mathrm{T} = e^{i\pi/8}\begin{bmatrix} e^{-i\pi/8} & 0 \\ 0 & e^{i\pi/8} \end{bmatrix} \tag{8}$$

A more generalised form of this gate [4] is used for rotation about a plane. These are given as follows,

$$\begin{bmatrix} \cos\frac{\gamma}{2} & -\sin\frac{\gamma}{2} \\ \sin\frac{\gamma}{2} & \cos\frac{\gamma}{2} \end{bmatrix} \tag{9}$$

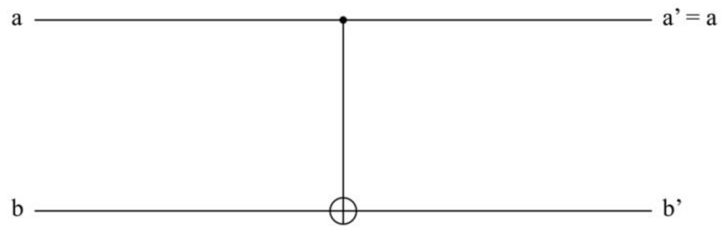
Multiple qubit gates

All quantum gates may be made to an arbitrary degree of precision by single or two qubit gates alone. However, the nomenclature "multiple" is used for the general purpose. The most widely used multiple qubit quantum gates are given below.

Controlled-NOT Gate or CNOT Gate [4] has two input qubits i.e. the control qubit and the target qubit. The logical implementation of this gate states, if the control qubit is 0 then the target qubit is left as it is and if the control qubit is set to 1, then the target qubit gets flipped. Mathematically,

$$|00\rangle \rightarrow |00\rangle \; ; |01\rangle \rightarrow |01\rangle \; ; |10\rangle \rightarrow |11\rangle \; ; |11\rangle \rightarrow |10\rangle \tag{10}$$

The circuit of CNOT Gate can be seen is Figure 2. The matrix representation of the CNOT Gate is as follows,



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Figure 2 : Quantum circuit representation for CNOT Gate

The truth table form of the above gate represents the logical XOR Gate in classical computation.

Controlled-CNOT Gate or CCNOT Gate has three input qubits. Of the three, two inputs are the control qubits and the other is the target qubit. The circuit for the CCNOT gate can be seen in Figure 3. Mathematically,

$$\begin{aligned}&|000\rangle \rightarrow |000\rangle \, ; |001\rangle \rightarrow |001\rangle \, ; |010\rangle \rightarrow |010\rangle \, ; |011\rangle \rightarrow |011\rangle \\ &|100\rangle \rightarrow |100\rangle \, ; |101\rangle \rightarrow |101\rangle \, ; |110\rangle \rightarrow |111\rangle \, ; |111\rangle \rightarrow |110\rangle\end{aligned} \tag{11}$$

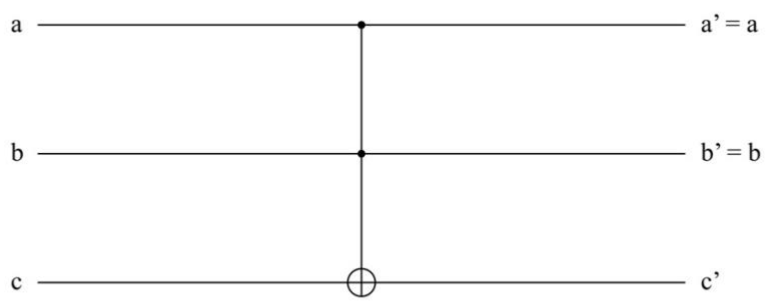


Figure 3 : Quantum circuit representation for CCNOT Gate

Swap Gate, as the name suggests, has two input qubit and it swaps the states without entanglement. The circuit for the gate can be seen in the Figure 4. Mathematically,

$$U_s| \psi, \varphi\rangle = U_s\,(|\psi \otimes \varphi\rangle)= |\varphi, \psi\rangle,$$

$$|00\rangle \rightarrow |00\rangle \, ; |01\rangle \rightarrow |10\rangle \, ; |10\rangle \rightarrow |01\rangle \, ; |11\rangle \rightarrow |11\rangle \tag{12}$$

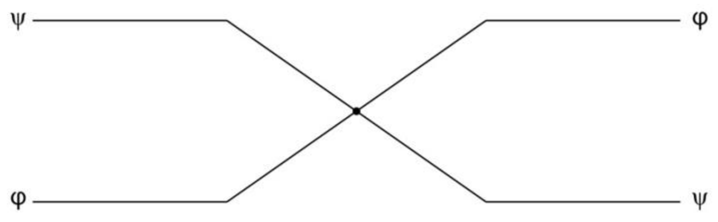


Figure 4 : Quantum circuit representation for SWAP Gate

Controlled Swap Gate, has three inputs, that is, a control qubit and two target qubits. When the control qubit is 0, the states are left alone and if the control qubit is 1 then the two states are swapped without entanglement. The pictorial representation can be seen in Figure 5. Mathematically,

$$\begin{gathered}|000\rangle \rightarrow |000\rangle \;;\; |001\rangle \rightarrow |001\rangle \;;\; |010\rangle \rightarrow |010\rangle \;;\; |011\rangle \rightarrow |011\rangle \\ |100\rangle \rightarrow |100\rangle \;;\; |101\rangle \rightarrow |110\rangle \;;\; |110\rangle \rightarrow |101\rangle \;;\; |111\rangle \rightarrow |111\rangle\end{gathered} \tag{13}$$

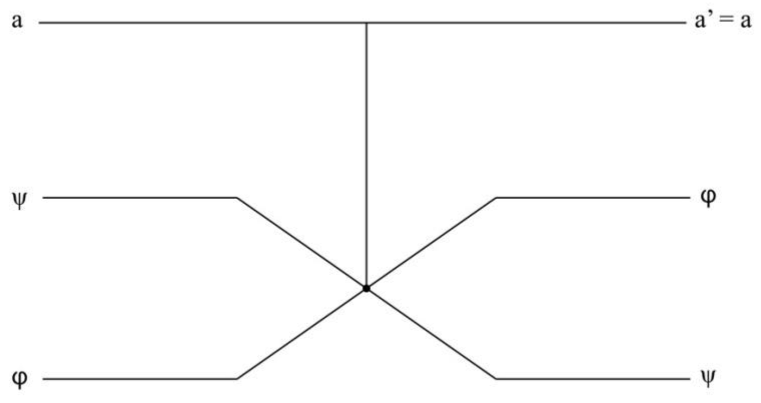


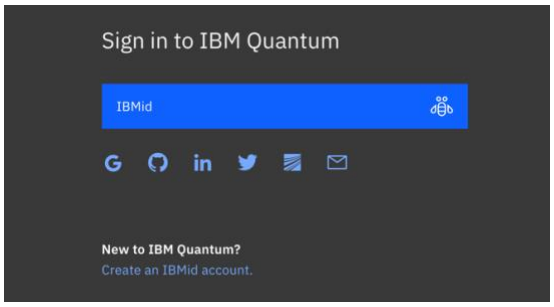
Figure 5 : Quantum circuit representation for CSWAP Gate

For further details on quantum mechanics, mathematical concepts and proofs pertaining to the subject can be seen in these standard textbooks [4] [10] [11] [12].

IBM Quantum Experience

IBM Quantum Experience is a compilation of IBM quantum composer and IBM Quantum Lab which form a cloud based quantum computing service provided by IBM. This lets us access IBM's prototype quantum processors. IBM quantum processors consist superconducting transmon qubits which are located in a dilution refrigerator at the IBM Headquarters.

Users initially use the URL [13] to reach the home page as shown in Figure. Now, the user must create an IBMid using Gmail, GitHub, LinkedIn, Twitter, Fraunhofer ID or their Email ID.



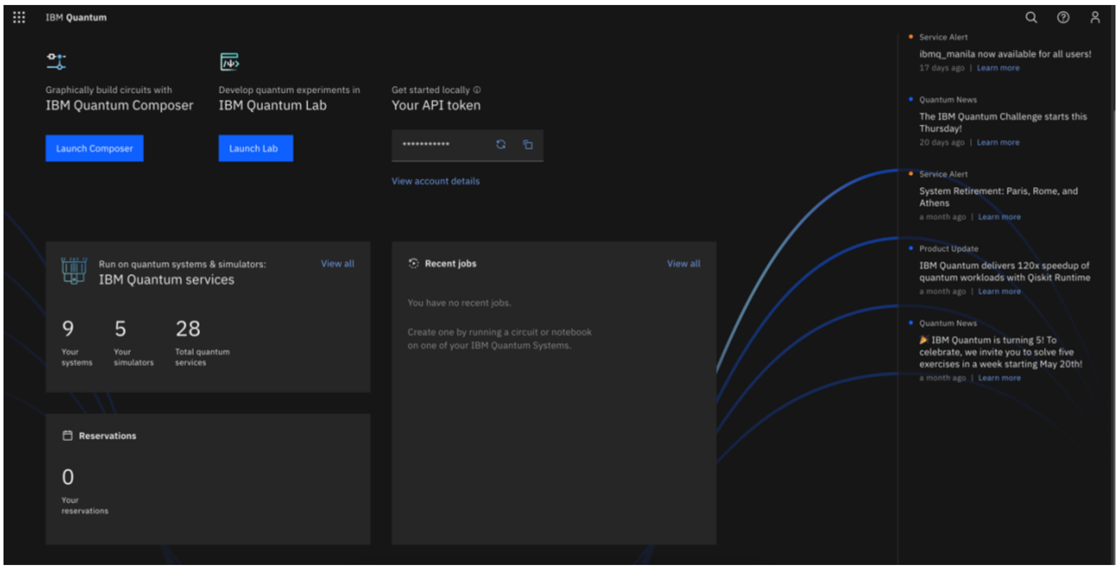
Sign in to IBM Quantum

IBMid

New to IBM Quantum?
Create an IBMid account.

Figure 6 : IBM Quantum Experience Login Prompt

It is advised to create a login with Gmail for smoother access. Once the login is created, the user is prompted to accept the terms and conditions of IBM Quantum Experience, which the user must agree to. Then they are prompted to enter their details, which must filled correctly. We are then presented with the welcome screen in Figure. From here, the user must click on Launch Composer to execute as was done in this research. But, they may also use the Launch Lab option according to their approach.



IBM Quantum

Graphically build circuits with
IBM Quantum Composer
Launch Composer

Develop quantum experiments in
IBM Quantum Lab
Launch Lab

Get started locally
Your API token
••••••••••••
View account details

Run on quantum systems & simulators:
IBM Quantum services
View all
9 Your systems
5 Your simulators
28 Total quantum services

Recent jobs
View all
You have no recent jobs.
Create one by running a circuit or notebook on one of your IBM Quantum Systems.

Reservations
0 Your reservations

Service Alert
ibmq_manila now available for all users!
17 days ago | Learn more

Quantum News
The IBM Quantum Challenge starts this Thursday!
20 days ago | Learn more

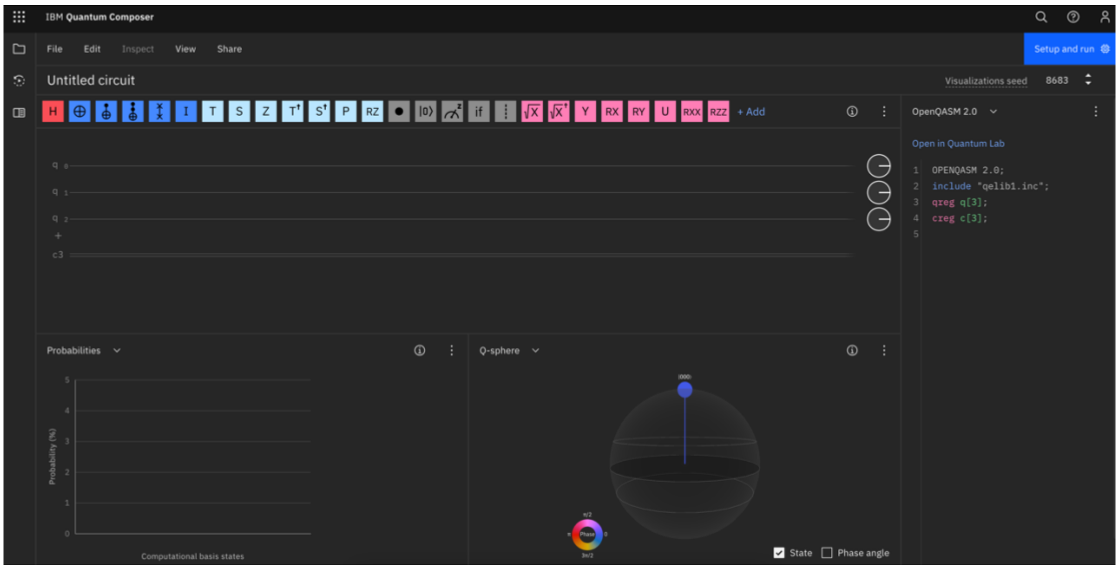
Service Alert
System Retirement: Paris, Rome, and Athens
a month ago | Learn more

Product Update
IBM Quantum delivers 120x speedup of quantum workloads with Qiskit Runtime
a month ago | Learn more

Quantum News
🎉 IBM Quantum is turning 5! To celebrate, we invite you to solve five exercises in a week starting May 20th!
a month ago | Learn more

Figure 7 : IBM Quantum Experience Home Screen

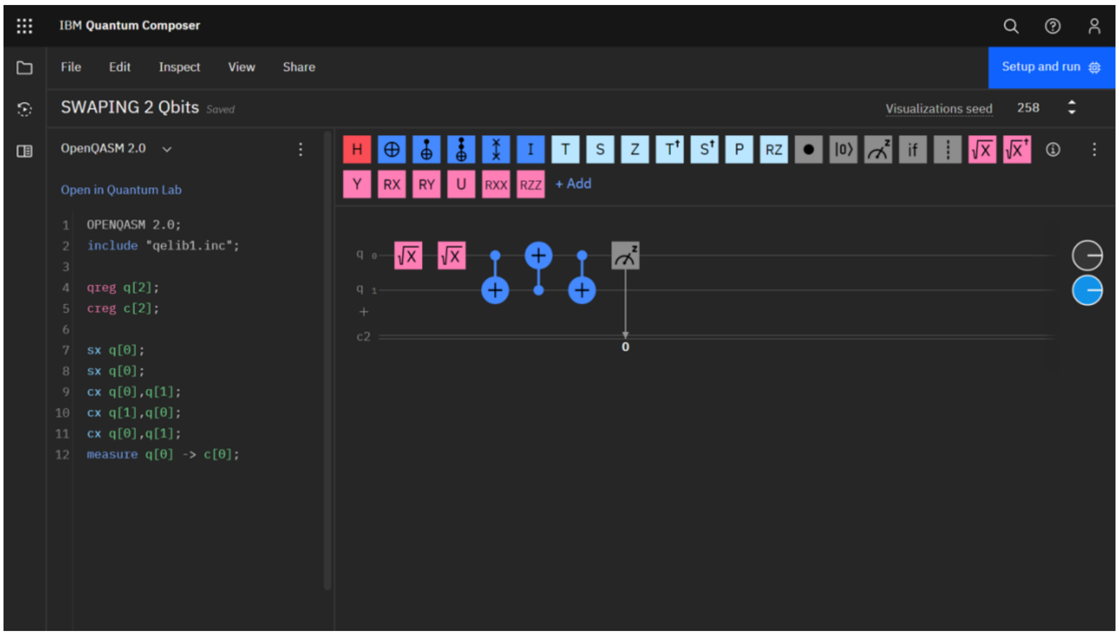
Once Launch Composer in clicked, the user is then redirected to the interface of the IBM Quantum Composer as in Figure. A block representation for all the quantum gates can be seen. These gates are then used to create a mathematically acceptable quantum circuit. It may also be observed that as we are designing our circuit, the code on the extreme right gets updated with every step or vice versa.



```
OPENQASM 2.0;
include "qelib1.inc";
qreg q[3];
creg c[3];
```

Figure 8 : IBM Quantum Composer

For example, in Figure 9, a X Gate has been realised using the $\sqrt{X}$ by multiplying it 2 times. And then creating a Swap Gate has been demonstrated using 3 CNOT Gates. In IBM Quantum Composer, all measurements can only be made using the measure block which has been used as demonstrated.



```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[2];
creg c[2];

sx q[0];
sx q[0];
cx q[0],q[1];
cx q[1],q[0];
cx q[0],q[1];
measure q[0] -> c[0];
```

Figure 9 : Basic quantum circuit on IBM Quantum Composer

Mathematically, a X Gate is used to flip the qubits. Here, q_0 and q_1 are always initialised to $|0\rangle$ by default. After passing through the X Gate,

$$q_0 = |1\rangle$$
$$q_1 = |0\rangle \qquad (a)$$

Now, a Swap Gate has been realised using 2 CNOT Gates. This can observed mathematically. Let us pass the qubits through the 1st CNOT Gate,

$$q_0 = |1\rangle$$
$$q_1 = |1\rangle \qquad (b)$$

Now, passing through the 2nd CNOT Gate,

$$q_0 = |0\rangle$$
$$q_1 = |1\rangle \qquad (c)$$

It can be intuitively observed that the 3rd CNOT Gate is just a redundancy in the circuit. But, the purpose is to demonstrate is that, if q_0 and q_1 are always initialised to $|0\rangle$ and $|1\rangle$, then the last CNOT Gate would come into the picture and complete the swapping. But, when the inputs are as (a), we would get the output after the 2nd CNOT Gate.

It can be observed that the interface is highly user friendly as for every gate, the code and hence the algorithm is formed simultaneously in openQASM 2.0 which is the abbreviation for Open Quantum Assembly Language.

```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[2];
creg c[2];

sx q[0];
sx q[0];
cx q[0],q[1];
cx q[1],q[0];
cx q[0],q[1];
```

It can be easily decoded that ‘sx’ is the keyword for the $\sqrt{X}$ Gate and similarly, ‘cx’ is the keyword for CNOT Gate. ‘q[n]’ means the nth qubit is the input for the gate. A much deeper analysis will help the user learn all the keywords and ultimately typing the code provided the rules and logic of quantum circuits holds good.

Shor’s Error Correcting Codes

In 1995, Peter Shor [6] demonstrated that quantum error correction holds extreme importance in the field of quantum computation. He first implemented a three-qubit error correcting code and went ahead to discover what is famously known as Shor’s nine-qubit error correcting code.

2.1 THREE QUBIT ERROR CORRECTING CODE

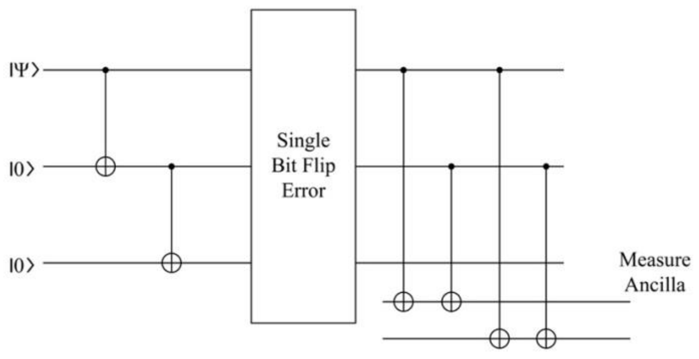


Figure 10 : Quantum circuit for Shor’s 3-qubit error correcting code

Figure 6 represents the Quantum circuit for Shor’s three-qubit error correcting code. The left half executes the encoding part while the right half executes the decoding. The encoding is as expressed in (14).

$$\alpha|0\rangle + \beta|1\rangle \rightarrow \alpha|000\rangle + \beta|111\rangle \tag{14}$$

Now, a noisy channel flips a qubit with a probability ‘p’ and leaves a qubit unchanged with a probability ‘(1-p)’. 0, 1, 2 or 3 flips are possible. Table 1 clearly elaborates these possibilities.

Flips	State	Probability
0	$\alpha|000\rangle + \beta|111\rangle$	$(1-p)^3$
1	$\alpha|100\rangle + \beta|011\rangle$ $\alpha|010\rangle + \beta|101\rangle$ $\alpha|001\rangle + \beta|110\rangle$	$3p(1-p)^2$
2	$\alpha|110\rangle + \beta|001\rangle$ $\alpha|101\rangle + \beta|010\rangle$ $\alpha|011\rangle + \beta|100\rangle$	$3p^2(1-p)$
3	$\alpha|111\rangle + \beta|000\rangle$	p^3

Table 1 : Probability of bit-flips in 3-qubit code

It is immediately clear that one of the above states would be input to the decoding circuit at the receiver. For instance, let us take the case with 0 flips and 1 flip,

$$\begin{aligned}&(\alpha|000\rangle + \beta|111\rangle)\,|00\rangle \\ &= \alpha|000\rangle|00\rangle + \beta|111\rangle|00\rangle \\ &= (\alpha|000\rangle + \beta|111\rangle)\,|00\rangle \end{aligned} \tag{15}$$

$$\begin{aligned}&(\alpha|100\rangle + \beta|011\rangle)\,|00\rangle \\ &= \alpha|100\rangle|11\rangle + \beta|011\rangle|11\rangle \\ &= (\alpha|100\rangle + \beta|011\rangle)|11\rangle \end{aligned} \tag{16}$$

In the first case (15), the ancilla is not affected. But when a qubit was flipped (16), there was a change in ancilla. The possible values of the ancilla are $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$. This change can be measured and a corrective action can be taken to make the information error free.

2.1 NINE QUBIT ERROR CORRECTING CODE

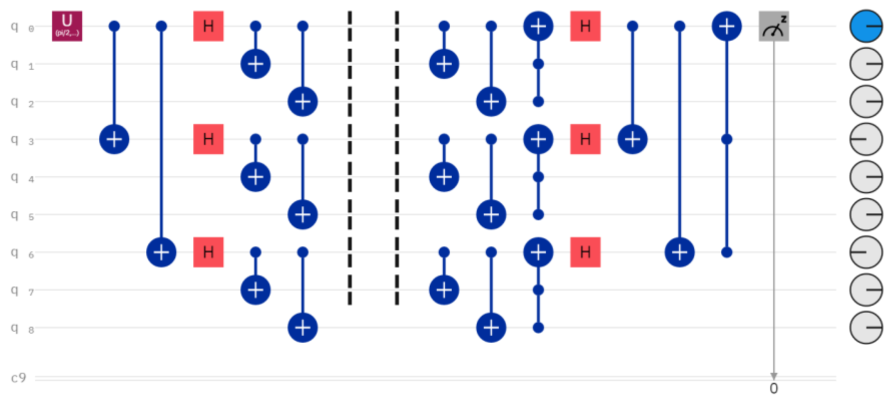


Figure 11 : Quantum circuit for Shor's 9-qubit error correcting code on IBM Environment

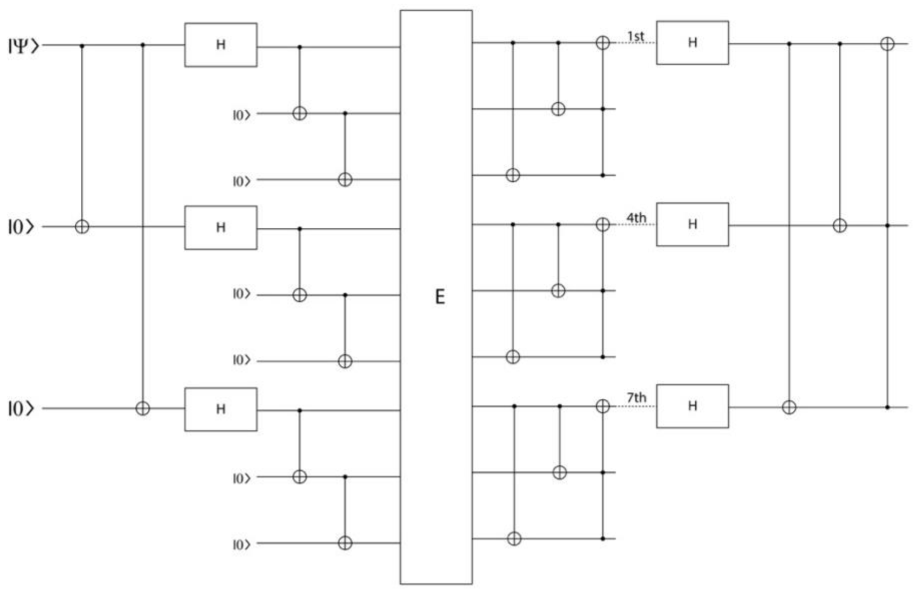


Figure 12 : Quantum circuit for Shor's 9-qubit error correcting code

The given quantum circuit in the Figure represents Shor's three qubit error correcting code. The left half executes the encoding part while the right half executes the decoding. Firstly, a phase error is converted into bit flip error. Phase error is due to the rotation of a qubit by an arbitrary angle φ. For simplicity, let φ = π, that is,

$$a|0\rangle + b|1\rangle \rightarrow a|0\rangle - b|1\rangle$$

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}} \rightarrow \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

$$|+\rangle \rightarrow |-\rangle \tag{17}$$

It may also be observed that the above special case in a simply a bit flip in the diagonal basis. Equation (17) simply demonstrates that phase flip in one basis is a bit flip in another basis.

For correcting phase errors, the qubits should be encoded as triplets in the diagonal basis as follows,

$$\psi = a|+\,+\,+\rangle + b|-\,-\,-\rangle \tag{18}$$

In the computational basis the state is as follows,

$$\frac{a+b}{2\sqrt{2}}\left[\,|000\rangle + |011\rangle + |101\rangle + |110\rangle\,\right] + \frac{a-b}{2\sqrt{2}}\left[\,|001\rangle + |010\rangle + |100\rangle + |111\rangle\,\right] \tag{19}$$

A measurement of any qubit, let us assume 1, in the computational basis will collapse to one of the following states,

$$\frac{a+b}{2\sqrt{2}}\left[\,|000\rangle + |011\rangle\,\right] + \frac{a-b}{2\sqrt{2}}\left[\,|001\rangle + |010\rangle\,\right] \tag{20}$$

$$\frac{a+b}{2\sqrt{2}}\left[\,|101\rangle + |110\rangle\,\right] + \frac{a-b}{2\sqrt{2}}\left[\,|100\rangle + |111\rangle\,\right] \tag{21}$$

A parity check is done on any if the collapsed states in equation (20) or equation (21), let us assume equation (20), in the diagonal basis by introducing two ancilla and measuring the ancilla after a CNOT operation as done in the 3-qubit code.

The above state collapses to the following state in the diagonal basis,

$$[\,a|+\,+\,+\rangle + b|-\,-\,-\rangle] + [\,a|-\,+\,+\rangle + b|+\,-\,-\rangle] \tag{22}$$

The above equation concludes that a state is either without error or a single bit flip error which can be removed after corrective action after the measurement of the ancilla.

Now, moving forward with knowledge of conversion of phase flips to bit flips and their mathematics in the diagonal and computational basis, Shor's 9-qubit error code takes the case of,

a) Bit flip error : X $\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} b \\ a \end{bmatrix}$

b) Phase flip error : Z $\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a \\ -b \end{bmatrix}$

c) Phase and Bit flip error : Y $\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} -b \\ a \end{bmatrix}$

Let the input state be a$|0\rangle$ + b$|1\rangle$ and the qubits $|0\rangle$ and $|1\rangle$ be encoded as triplets in the diagonal basis as follows,

$$|0\rangle \rightarrow |0\rangle_L \rightarrow |+ + +\rangle \tag{23}$$

$$|1\rangle \rightarrow |1\rangle_L \rightarrow | - - - \rangle \tag{24}$$

Mathematically,

$$(a|0\rangle + b|1\rangle)\ |0\rangle\ |\ 0\rangle$$

$$a|000\rangle + b|111\rangle \tag{25}$$

Applying Hadamard Gate,

$$\frac{a}{2\sqrt{2}}\ [\ (|0\rangle + |1\rangle)\ (|0\rangle + |1\rangle)\ (|0\rangle + |1\rangle)] + \frac{b}{2\sqrt{2}}\ [\ (|0\rangle - |1\rangle)\ (|0\rangle - |1\rangle)\ (|0\rangle - |1\rangle)\ ] \tag{26}$$

Now multiplying with $|00\rangle$,

$$\frac{a}{2\sqrt{2}}\ [\ (|000\rangle + |111\rangle)\ (|000\rangle + |111\rangle)\ (|000\rangle + |111\rangle)\ ] + \frac{b}{2\sqrt{2}}\ [\ (|000\rangle - |111\rangle)\ (|000\rangle - |111\rangle)\ (|000\rangle - |111\rangle)\ ]$$

$$\Rightarrow a|+ + +\rangle + b|- - -\rangle \tag{27}$$

Out of the nine qubits, at most one qubit may be affected. In most practical cases, the probability is 0.01 that a qubit may flip. Assuming this to be the scenario while decoding, the above states change where the first qubit is flipped (for simplicity) as following,

$$\frac{a}{2\sqrt{2}}\,[\,(|100\rangle + |011\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)\,] + \frac{b}{2\sqrt{2}}\,[\,(|100\rangle - |011\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)\,] \tag{28}$$

Passing the above states through CCNOT Gate,

$$\frac{a}{2\sqrt{2}}\,[\,(|011\rangle - |111\rangle)(|000\rangle + |100\rangle)(|000\rangle + |100\rangle)\,] + \frac{b}{2\sqrt{2}}\,[\,(|011\rangle + |111\rangle)(|000\rangle - |100\rangle)(|000\rangle - |100\rangle)\,]$$

$$\Rightarrow \frac{a}{2\sqrt{2}}\,[\,(|0\rangle - |1\rangle)|11\rangle\,(|0\rangle + |1\rangle)|00\rangle\,(|0\rangle + |1\rangle)\,|00\rangle\,] + \frac{b}{2\sqrt{2}}\,[\,(|0\rangle + |1\rangle)|11\rangle\,(|0\rangle - |1\rangle)|00\rangle\,(|0\rangle - |1\rangle)\,|00\rangle\,] \tag{29}$$

Now, the 1st, 4th and 7th bits are passed through Hadamard Gates,

$$\mathrm{a}|1\rangle|0\rangle|0\rangle + \mathrm{b}|0\rangle|1\rangle|1\rangle \tag{30}$$

Applying CNOT with 4th and 7th bit as control,

$$\mathrm{a}|1\rangle|1\rangle|1\rangle + \mathrm{b}|0\rangle|1\rangle|1\rangle$$

$$(\mathrm{a}|0\rangle + \mathrm{b}|1\rangle)|11\rangle \tag{31}$$

It can be deduced from equation (31) that we are successfully able to decode and receive the original qubit states that we sent. A similar logic has been applied and is further discussed in the upcoming section.

A Novel 12-Qubit Multiple Quantum Error Correcting Code

Now consider a wavefunction $\psi = \alpha|0\rangle + \beta|1\rangle$ where the probability of being in state $|0\rangle$ is $|\alpha|^2$ and probability of being in state $|1\rangle$ is $|\beta|^2$.

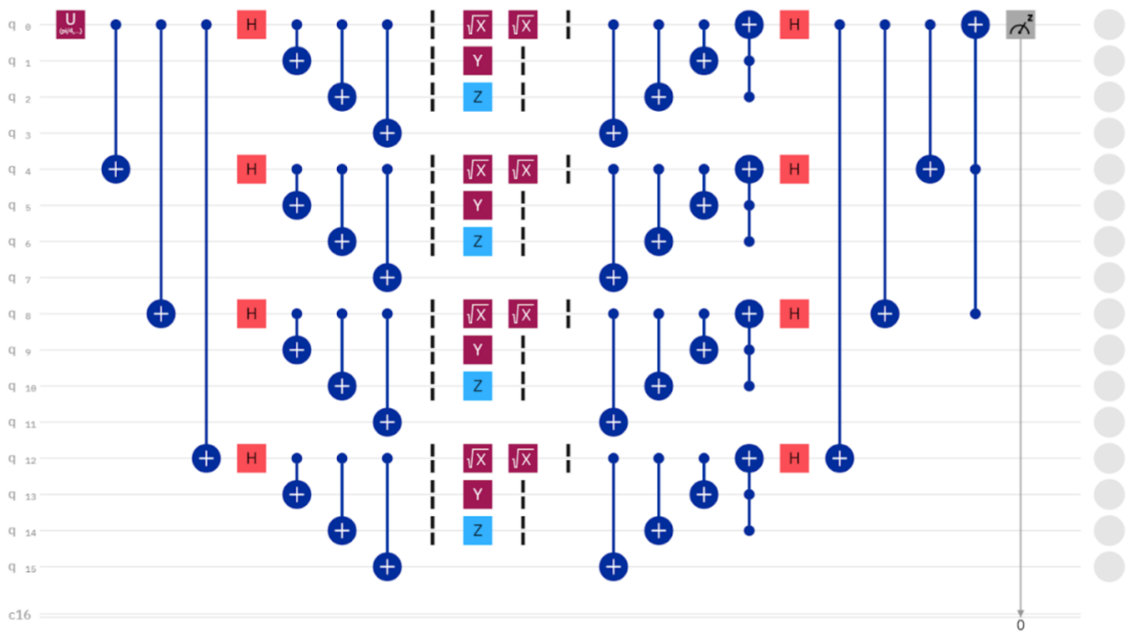


Figure 13 : Quantum circuit for 12-qubit error correcting code

3.1 ENCODING CIRCUIT

At the first phase of the encoding circuit shows the qubit flip error and then following after Hadamard gates shows the qubit phase flip error in this way we can see the designed circuit is capable of dealing both phase flip and bit flip errors. Ancilla bits are generally considered as garbage bits in Quantum computation because their effects are not considered in the computation and they can be reused, ancilla bits are very vital in quantum algorithms, since we can't clone the states of the qubit (No-cloning-theorem), we use ancilla bits in repetition codes.

Applying Ancilla bits q_4, q_8, q_{12}, on state ψ gives us,

$$\alpha|0_0 0_4 0_8 0_{12}\rangle + \beta|1_0 1_4 1_8 1_{12}\rangle \tag{32}$$

Applying Hadamard Gates $[\ \alpha|0_0\rangle + \beta|1_0\rangle\ ] \otimes 4H$,

$$\Rightarrow \frac{\alpha}{4}\left[(|0_0\rangle + |1_0\rangle)|0_1 0_2 0_3\rangle(|0_4\rangle + |1_4\rangle)|0_5 0_6 0_7\rangle(|0_8\rangle + |1_8\rangle)|0_9 0_{10} 0_{11}\rangle(|0_{12}\rangle + |1_{12}\rangle)|0_{13} 0_{14} 0_{15}\rangle\right]$$
$$+ \frac{\beta}{4}\left[(|0_0\rangle - |1_0\rangle)|0_1 0_2 0_3\rangle(|0_4\rangle - |1_4\rangle)|0_5 0_6 0_7\rangle(|0_8\rangle - |1_8\rangle)|0_9 0_{10} 0_{11}\rangle(|0_{12}\rangle - |1_{12}\rangle)|0_{13} 0_{14} 0_{15}\rangle\right] \tag{33}$$

$$\Rightarrow \frac{\alpha}{4}\left[(|0_0 0_1 0_2 0_3\rangle + |1_0 1_1 1_2 1_3\rangle)(|0_4 0_5 0_6 0_7\rangle + |1_4 1_5 1_6 1_7\rangle)(|0_8 0_9 0_{10} 0_{11}\rangle + |1_8 1_9 1_{10} 1_{11}\rangle)(|0_{12} 0_{13} 0_{14} 0_{15}\rangle + |1_{12} 1_{13} 1_{14} 1_{15}\rangle)\right]$$
$$+ \frac{\beta}{4}\left[(|0_0 0_1 0_2 0_3\rangle - |1_0 1_1 1_2 1_3\rangle)(|0_4 0_5 0_6 0_7\rangle - |1_4 1_5 1_6 1_7\rangle)(|0_8 0_9 0_{10} 0_{11}\rangle - |1_8 1_9 1_{10} 1_{11}\rangle)(|0_{12} 0_{13} 0_{14} 0_{15}\rangle - |1_{12} 1_{13} 1_{14} 1_{15}\rangle)\right] \tag{34}$$

Assume σ_y error has occurred on 0th qubit due to noisy channel, there will be a bit flip and phase error on 0th qubit,

$$\Rightarrow \frac{\alpha}{4}\left[(|1_0 0_1 0_2 0_3\rangle - |0_0 1_1 1_2 1_3\rangle)(|0_4 0_5 0_6 0_7\rangle + |1_4 1_5 1_6 1_7\rangle)(|0_8 0_9 0_{10} 0_{11}\rangle + |1_8 1_9 1_{10} 1_{11}\rangle)(|0_{12} 0_{13} 0_{14} 0_{15}\rangle + |1_{12} 1_{13} 1_{14} 1_{15}\rangle)\right]$$
$$+ \frac{\beta}{4}\left[(|1_0 0_1 0_2 0_3\rangle + |0_0 1_1 1_2 1_3\rangle)(|0_4 0_5 0_6 0_7\rangle - |1_4 1_5 1_6 1_7\rangle)(|0_8 0_9 0_{10} 0_{11}\rangle - |1_8 1_9 1_{10} 1_{11}\rangle)(|0_{12} 0_{13} 0_{14} 0_{15}\rangle - |1_{12} 1_{13} 1_{14} 1_{15}\rangle)\right] \tag{35}$$

3.2 DECODING CIRCUIT

While coming to the other side of the circuit after crossing the noisy channel, the qubits are first sent to CNOT. Applying CNOT as 0th , 4th ,8th ,12th qubit as control and 1st, 2nd, 3rd, 5th, 6th, 7th, 9th, 10th, 11th, 13th, 14th, 15th qubits are target bits,

$$\Rightarrow \frac{\alpha}{4}\left[(|1_0 1_1 1_2 1_3\rangle - |0_0 1_1 1_2 1_3\rangle)(|0_4 0_5 0_6 0_7\rangle + |1_4 0_5 0_6 0_7\rangle)(|0_8 0_9 0_{10} 0_{11}\rangle + |1_8 0_9 0_{10} 0_{11}\rangle)(|0_{12} 0_{13} 0_{14} 0_{15}\rangle + |1_{12} 0_{13} 0_{14} 0_{15}\rangle)\right]$$
$$+ \frac{\beta}{4}\left[(|1_0 1_1 1_2 1_3\rangle + |0_0 1_1 1_2 1_3\rangle)(|0_4 0_5 0_6 0_7\rangle - |1_4 0_5 0_6 0_7\rangle)(|0_8 0_9 0_{10} 0_{11}\rangle - |1_8 0_9 0_{10} 0_{11}\rangle)(|0_{12} 0_{13} 0_{14} 0_{15}\rangle - |1_{12} 0_{13} 0_{14} 0_{15}\rangle)\right] \tag{36}$$

Applying CCNOT gate,

$$\Rightarrow \frac{\alpha}{4}\left[(|0_0 1_1 1_2 1_3\rangle - |1_0 1_1 1_2 1_3\rangle)(|0_4 0_5 0_6 0_7\rangle + |1_4 0_5 0_6 0_7\rangle)(|0_8 0_9 0_{10} 0_{11}\rangle + |1_8 0_9 0_{10} 0_{11}\rangle)(|0_{12} 0_{13} 0_{14} 0_{15}\rangle + |1_{12} 0_{13} 0_{14} 0_{15}\rangle)\right]$$
$$+ \frac{\beta}{4}\left[(|0_0 1_1 1_2 1_3\rangle + |1_0 1_1 1_2 1_3\rangle)(|0_4 0_5 0_6 0_7\rangle - |1_4 0_5 0_6 0_7\rangle)(|0_8 0_9 0_{10} 0_{11}\rangle - |1_8 0_9 0_{10} 0_{11}\rangle)(|0_{12} 0_{13} 0_{14} 0_{15}\rangle - |1_{12} 0_{13} 0_{14} 0_{15}\rangle)\right] \tag{37}$$

$$\Rightarrow \frac{\alpha}{4}\left[(|0_0\rangle - |1_0\rangle)|1_1 1_2 1_3\rangle(|0_4\rangle + |1_4\rangle)|0_5 0_6 0_7\rangle(|0_8\rangle + |1_8\rangle)|0_9 0_{10} 0_{11}\rangle(|0_{12}\rangle + |1_{12}\rangle)|0_{13} 0_{14} 0_{15}\rangle\right]$$
$$+ \frac{\beta}{4}\left[(|0_0\rangle + |1_0\rangle)|1_1 1_2 1_3\rangle(|0_4\rangle - |1_4\rangle)|0_5 0_6 0_7\rangle(|0_8\rangle - |1_8\rangle)|0_9 0_{10} 0_{11}\rangle(|0_{12}\rangle - |1_{12}\rangle)|0_{13} 0_{14} 0_{15}\rangle\right] \tag{38}$$

Hadamard gates are very important for correcting phase flip errors. Applying on 0th, 4th, 8th, 12th gives,

$$\alpha|1_0\rangle|0_4\rangle|0_8\rangle|0_{12}\rangle + \beta|0_0\rangle|1_4\rangle|1_8\rangle|1_{12}\rangle \tag{39}$$

Applying CNOT as 1st qubit as control and 4th, 8th, 12th qubits as target bits, since the 1st qubit is 1 it toggles the target bit,

$$\alpha|1_0\rangle|1_4\rangle|1_8\rangle|1_{12}\rangle + \beta|0_0\rangle|1_4\rangle|1_8\rangle|1_{12}\rangle \tag{40}$$

Applying CCNOT,

$$\alpha|0_0\rangle|1_4\rangle|1_8\rangle|1_{12}\rangle + \beta|1_0\rangle|1_4\rangle|1_8\rangle|1_{12}\rangle \tag{41}$$

$$(\alpha|0_0\rangle + \beta|1_0\rangle)|1_4\rangle|1_8\rangle|1_{12}\rangle \tag{42}$$

Thus we have obtained the original wave function as following,

$$\psi = \alpha|0_0\rangle + \beta|1_0\rangle \tag{43}$$

Now apply an general single qubit rotation gate with 3 Euler angles U(θ, ϕ, λ) applied on ψ , where $\theta = \pi/4$, $\phi = \pi/2$, $\lambda = \pi/2$ gives,

$$\psi = \sqrt{0.85}|0_0\rangle + \sqrt{0.15}|1_0\rangle \tag{44}$$

It can be observed that, we receive the bits sent as is but the first and second qubit are received with the probability of 0.85 and 0.15 respectively.

Results

For testing and verification of the 12-qubit error correcting, 4 IBM Quantum simulators were used. Qubits in these devices are made up of superconducting transmons. Most of the devices are of less quantum volume ranging 5 to 15 qubits. There are only a few simulators which have a range of 32 to 64 qubits. The list of devices used are given in table 2. Frequency versus State of the particle for different simulators used in IBM QX are shown in Figure 14.

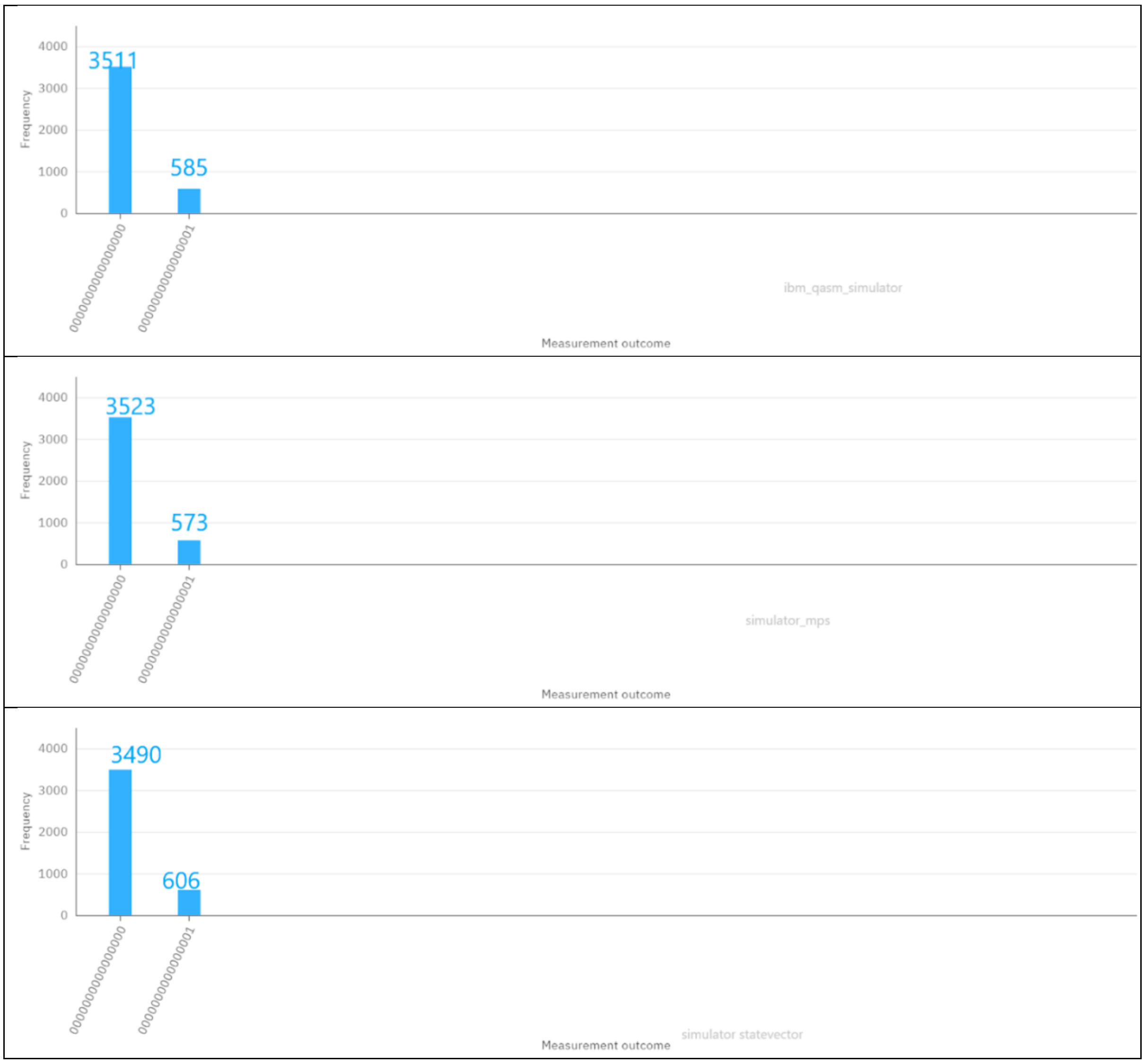


Figure 14 : Frequency versus State

Name	Qubit Volume	Expected Value	Actual Value	Relative Error
ibmq_qasm_simulator	32	$P(\|0_0\rangle) = 0.850000$ $P(\|1_0\rangle) = 0.150000$	$P(\|0_0\rangle) = 0.857177$ $P(\|1_0\rangle) = 0.142822$	0.844 % 4.79 %
simulator_statevector	32	$P(\|0_0\rangle) = 0.850000$ $P(\|1_0\rangle) = 0.150000$	$P(\|0_0\rangle) = 0.852050$ $P(\|1_0\rangle) = 0.147949$	0.241 % 1.367 %
simulator_mps	100	$P(\|0_0\rangle) = 0.850000$ $P(\|1_0\rangle) = 0.150000$	$P(\|0_0\rangle) = 0.860107$ $P(\|1_0\rangle) = 0.0139892$	1.19 % 6.74 %

Table 2 : List of devices and computed values

Conclusion

In this paper, we designed a novel 12-qubit Multiple error correcting codes using the qiskit library provided by IBM, to record the performance of the design we ran the circuit 4096 times each on three different IBM quantum simulators. We believe that this approach can be further extended to implement 15, 18 or 21 i.e. up to 3n qubits. We achieved correcting multiple bit errors taking an extra qubit. Using CNOT, CCNOT, Pauli-X, Pauli-Y, Pauli-Z and Hadamard Gates, the quantum cost of this method may increase a bit, but owing to the high qubit precision the percentage of error reduces significantly. A close study clearly demonstrates how closely related the concepts of classical error correction and quantum error correction related with their core principle being adding a redundancy to preserve the originality of the information being transmitted and processed. Further research in this line of work may reveal itself to be fruitful as quantum computation solves many time complexity problems faced with classical computation and quantum error correction would prove to be of vital importance.

References

1) Warke, B. K. Behera, P. K. Panigrahi, “*The first three-qubit and six-qubit full quantum multiple error-correcting codes with low quantum costs*,” Research Gate, July 2019
2) H. Chang, *An Introduction to Error-Correcting Codes: From Classical to Quantum*, arXiv Quantum Physics, Cornell University, February 2006.
3) W. K. Wooters and W. H. Zurek, “*A single quantum can- not be cloned*,” Nature, vol. 299, pp. 802–803, 1982.
4) M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge, UK: Cambridge University Press, 2000.
5) B. Schumacher, “*Quantum coding*,” Phys. Rev. A, vol. 51, pp. 2738–2747, 1995
6) P. W. Shor, “*Scheme for reducing decoherence in quantum memory*,” Phys. Rev. A, vol. 52, no. 4, pp. R2493–R2496, 1995.
7) A. M. Steane, “*Error correcting codes in quantum theory*,” Phys. Rev. Lett., vol. 77, no. 5, pp. 793–797, 1996.
8) C. H. Bennett and P. W. Shor, “*Quantum information theory*,” IEEE Trans. Inform. Theory, vol. 44, no. 6, pp. 2724– 2742, 1998.
9) C. Shannon, “*A mathematical theory of communication*,” Bell Syst. Tech. J., vol. 27, pp. 379–423, 1948.
10) D. J. Griffiths, *Introduction to Quantum Mechanics*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 2005.
11) R. B. Griffiths, *Consistent Quantum Theory*, Cambridge, UK: Cambridge University Press, 2002.
12) R. L. Liboff, *Introductory Quantum Mechanics*, 4th ed. Francisco, CA: Addison Wesley, 2003.
13) IBM Quantum, https://quantum-computing.ibm.com/, 2021

Appendix

The following is the code generated by the IBM quantum simulator.

QASM:

```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[16];
creg c[16];

u(pi/4,pi/2,pi/2) q[0];
cx q[0],q[4];
cx q[0],q[8];
cx q[0],q[12];
h q[0];
h q[4];
h q[8];
h q[12];
cx q[0],q[1];
cx q[4],q[5];
cx q[8],q[9];
cx q[12],q[13];
cx q[0],q[2];
cx q[4],q[6];
cx q[8],q[10];
cx q[12],q[14];
cx q[0],q[3];
cx q[4],q[7];
cx q[8],q[11];
cx q[12],q[15];
barrier q[0];
barrier q[1];
barrier q[2];
barrier q[4];
barrier q[5];
barrier q[6];
barrier q[8];
barrier q[9];
barrier q[10];
barrier q[12];
barrier q[13];
barrier q[14];
sx q[0];
y q[1];
z q[2];
```

```
sx q[4];
y q[5];
z q[6];
sx q[8];
y q[9];
z q[10];
sx q[12];
y q[13];
barrier q[13];
z q[14];
sx q[0];
barrier q[1];
barrier q[2];
sx q[4];
barrier q[5];
barrier q[6];
sx q[8];
barrier q[9];
barrier q[10];
sx q[12];
barrier q[14];
barrier q[0];
barrier q[4];
barrier q[8];
barrier q[12];
cx q[0],q[3];
cx q[4],q[7];
cx q[8],q[11];
cx q[12],q[15];
cx q[0],q[2];
cx q[4],q[6];
cx q[8],q[10];
cx q[12],q[14];
cx q[0],q[1];
cx q[4],q[5];
cx q[8],q[9];
cx q[12],q[13];
ccx q[2],q[1],q[0];
ccx q[6],q[5],q[4];
ccx q[10],q[9],q[8];
ccx q[13],q[14],q[12];
h q[0];
h q[4];
h q[8];
h q[12];
cx q[0],q[12];
cx q[0],q[8];
cx q[0],q[4];
ccx q[4],q[8],q[0];
measure q[0] -> c[0];
```

QISKIT:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(16, 'q')
creg_c = ClassicalRegister(16, 'c')
circuit = QuantumCircuit(qreg_q, creg_c)

circuit.u(pi/4, pi/2, pi/2, qreg_q[0])
circuit.cx(qreg_q[0], qreg_q[4])
circuit.cx(qreg_q[0], qreg_q[8])
circuit.cx(qreg_q[0], qreg_q[12])
circuit.h(qreg_q[0])
circuit.h(qreg_q[4])
circuit.h(qreg_q[8])
circuit.h(qreg_q[12])
circuit.cx(qreg_q[0], qreg_q[1])
circuit.cx(qreg_q[4], qreg_q[5])
circuit.cx(qreg_q[8], qreg_q[9])
circuit.cx(qreg_q[12], qreg_q[13])
circuit.cx(qreg_q[0], qreg_q[2])
circuit.cx(qreg_q[4], qreg_q[6])
circuit.cx(qreg_q[8], qreg_q[10])
circuit.cx(qreg_q[12], qreg_q[14])
circuit.cx(qreg_q[0], qreg_q[3])
circuit.cx(qreg_q[4], qreg_q[7])
circuit.cx(qreg_q[8], qreg_q[11])
circuit.cx(qreg_q[12], qreg_q[15])
circuit.barrier(qreg_q[0])
circuit.barrier(qreg_q[1])
circuit.barrier(qreg_q[2])
circuit.barrier(qreg_q[4])
circuit.barrier(qreg_q[5])
circuit.barrier(qreg_q[6])
circuit.barrier(qreg_q[8])
circuit.barrier(qreg_q[9])
circuit.barrier(qreg_q[10])
circuit.barrier(qreg_q[12])
circuit.barrier(qreg_q[13])
circuit.barrier(qreg_q[14])
circuit.sx(qreg_q[0])
circuit.y(qreg_q[1])
circuit.z(qreg_q[2])
circuit.sx(qreg_q[4])
circuit.y(qreg_q[5])
circuit.z(qreg_q[6])
circuit.sx(qreg_q[8])
circuit.y(qreg_q[9])
circuit.z(qreg_q[10])
circuit.sx(qreg_q[12])
circuit.y(qreg_q[13])
circuit.barrier(qreg_q[13])
circuit.z(qreg_q[14])
circuit.sx(qreg_q[0])
circuit.barrier(qreg_q[1])
circuit.barrier(qreg_q[2])
circuit.sx(qreg_q[4])
circuit.barrier(qreg_q[5])
circuit.barrier(qreg_q[6])
circuit.sx(qreg_q[8])
circuit.barrier(qreg_q[9])
circuit.barrier(qreg_q[10])
circuit.sx(qreg_q[12])
circuit.barrier(qreg_q[14])
circuit.barrier(qreg_q[0])
circuit.barrier(qreg_q[4])
circuit.barrier(qreg_q[8])
circuit.barrier(qreg_q[12])
circuit.cx(qreg_q[0], qreg_q[3])
circuit.cx(qreg_q[4], qreg_q[7])
circuit.cx(qreg_q[8], qreg_q[11])
circuit.cx(qreg_q[12], qreg_q[15])
circuit.cx(qreg_q[0], qreg_q[2])
circuit.cx(qreg_q[4], qreg_q[6])
circuit.cx(qreg_q[8], qreg_q[10])
circuit.cx(qreg_q[12], qreg_q[14])
circuit.cx(qreg_q[0], qreg_q[1])
circuit.cx(qreg_q[4], qreg_q[5])
circuit.cx(qreg_q[8], qreg_q[9])
circuit.cx(qreg_q[12], qreg_q[13])
circuit.ccx(qreg_q[2], qreg_q[1], qreg_q[0])
circuit.ccx(qreg_q[6], qreg_q[5], qreg_q[4])
circuit.ccx(qreg_q[10], qreg_q[9], qreg_q[8])
circuit.ccx(qreg_q[13], qreg_q[14], qreg_q[12])
circuit.h(qreg_q[0])
circuit.h(qreg_q[4])
circuit.h(qreg_q[8])
circuit.h(qreg_q[12])
circuit.cx(qreg_q[0], qreg_q[12])
circuit.cx(qreg_q[0], qreg_q[8])
circuit.cx(qreg_q[0], qreg_q[4])
circuit.ccx(qreg_q[4], qreg_q[8], qreg_q[0])
circuit.measure(qreg_q[0], creg_c[0])
```