VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI - 590018

Report on Internship (18ECI85)

QUANTUM ERROR DETECTION MODEL

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF ENGINEERING
IN
ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by

SHISHEER S KAUSHIK
1BG18EC127

Internship Carried Out
at
QWORLD ASSOCIATION

Kesklinna linnaosa, Tartu mnt 67/1-13b, 10115, Tallinn, Estonia

Internal Guide

Dr. Rekha P

Associate Professor,
Dept. of ECE, BNMIT

External Guide 1 Mr.

Zeki Can Seskir

PhD Student [QT],

Karlsruhe Institute of Technical (KIT), Germany

External Guide 2
Mr. Adam Glos

Program Head & Coordinator (QWorld), Estonia

Vidyayāmruthamashnuthe

B.N.M. Institute of Technology

An Autonomous Institution under VTU, Approved by AICTE, Accredited as Grade A Institution by NAAC.
All Eligible UG branches – CSE, ECE, EEE, ISE & Mech.E Accredited by NBA for academic years 2018-19 to 2021-22 & valid up to 30.06.2022
Post box no. 7087, 27th cross, 12th Main, Banashankari 2nd Stage, Bengaluru- 560070, INDIA
Ph.: 91-80- 26711780/81/82 Email: principal@bnmit.in, www.bnmit.org

Department of Electronics & Communication Engineering

2021 – 2022

B.N.M. Institute of Technology

An Autonomous Institution under VTU, Approved by AICTE, Accredited as Grade A Institution by NAAC.
All Eligible UG branches – CSE, ECE, EEE, ISE & Mech.E Accredited by NBA for academic years 2018-19 to 2021-22 & valid up to 30.06.2022
Post box no. 7087, 27th cross, 12th Main, Banashankari 2nd Stage, Bengaluru- 560070, INDIA
Ph.: 91-80- 26711780/81/82 Email: principal@bnmit.in, www.bnmit.org

DEPARTMENT: ELECTRONICS & COMMUNICATION ENGINEERING

Vidyayāmruthamashnuthe

CERTIFICATE

This is to certify that the Internship entitled **Quantum Error Detection Model** carried out by **Mr. Shisheer S Kaushik** USN **1BG18EC127** a bonafide student of VIII Semester, in partial fulfillment for the Bachelor of Engineering in Electronics & Communication Engineering of the **Visvesvaraya Technological University**, Belagavi during the year 2021-22. It is certified that all corrections / suggestions indicated have been incorporated in the report. The internship report has been approved as it satisfies the academic requirements in respect of internship work prescribed for the said degree.

Dr. Rekha P	**Dr. P.A. Vijaya**	**Dr. Krishnamurthy G. N.**
Associate Professor,	**Professor and Head**	**Principal**
Dept. of ECE, BNMIT	**Dept. of ECE, BNMIT**	**BNMIT, Bengaluru**

External Viva Voce

Name of the Examiner	**Signature with Date**
1) _____	_____
2) _____	_____

QWORLD

DIPLOMA

presented to

Shisheer S Kaushik

for participating in the project **Mapping the Landscape of Quantum Education Efforts** led by Zeki Can Seskir organized during QIntern 2021

Adam Glos
(Adam Glos, QWorld)

Zoltán Zimborás
(Zoltán Zimborás, QWorld)

Diploma Number: QIntern2021-88
01.07.2021-22.08.2021

qworld.net | We are working to build an open quantum ecosystem

DECLARATION

I, the undersigned solemnly declare that the report of the internship work entitled **"Quantum Error Detection Model"** is based on my work carried out during the course study under the supervision of **Dr. Rekha P, Associate Professor, Department of ECE,** BNM Institute of Technology, Bangalore and **Mr. Adam Glos, Project Director** along with **Mr. Zeki Can Seskir, Mentor, PhD Student**, Karlsruhe Institute of Technical (KIT)**,** Germany. This Remote internship work was carried out at QIntern-21 organized by QWord Association, Global Network.

I assert that the statements made and conclusions drawn are an outcome of the internship work. I further declare that, to the best of my knowledge and belief, that this report does not contain any work which has been submitted for the award of the degree or any other degree in this university or any other university.

SHISHEER S KAUSHIK
1BG18EC127

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of the internship would be incomplete without mentioning the names of the people who made it possible.

I am greatly indebted to our Director, **Prof. T.J. Rama Murthy**, Additional Director**, Dr. S.Y.Kulkarni,** Dean, **Prof. Eishwar N. Maanay,** Principal, **Dr. Krishnamurthy G.N**, B.N.M Institute of Technology, Bangalore and **Dr. P.A. Vijaya,** Head, Dept. of, Electronics and Communication Engg, for giving me an opportunity to carry out the Internship. I am really indebted to our great institute BNMIT, which has provided me an opportunity to get a better degree in B.E. with the state- of-the-art facilities, and an impeccable environment.

It is with a deep sense of gratitude and great respect, I owe my indebtedness to my guide **Dr. Rekha P.** I thank for her constant encouragement during the execution of this Internship and also I thank my Internship coordinator **Smt. Keerti Kulkarni** for making the conduction of this Internship successful.

I express my heartfelt thanks to my External Guides at **Mr. Zeki Can Seskir, Mr. Adam Glos & their Team (Qworld Association**) for their guidance and unconditional support, despite of their busy work schedule. I thank my **parents**, without whom none of this would have been possible. Their patience and blessings have been with me at every step of this Internship. I would express my thanks to all my friends and all those who have helped me directly or indirectly for the successful completion of the Internship.

SHISHEER S KAUSHIK

1BG18EC127

EXECUTIVE SUMMARY

This report is prepared based on four weeks of internship program that I have done at QWorld Global Association as a requirement of my B.E., program in Visvesvaraya Technological University (VTU).

I was part of QIntern-21 program, which is organized annually by QWorld Global Association. Fortuitously, I got short listed through the online Interview and Technical Background Test. During these four weeks of internship, I was allotted to one respective group under a mentor/supervisor to work on the Major Computational Issue faced during Quantum Application Programming, which was already in progress, in this group I had a chance to learn a lot about **Quantum Error Detection**. It was a very effective learning experience. Throughout the Course of my Internship, I indulged in the development of the project named as '**Mapping the Landscape of Quantum Education Efforts'** under that, we were allotted to specific research lab with the respective mentors based on our interest. I opted to work in 'Quantum Error Correction Labs'. This internship has enhanced my skills and knowledge in this particular field via a hands on experience and also helped me to improve my network by meeting many reputed person working in this domain.

The work of My Project is divided as follows:

Part 1: gives an insight into how different types of errors affect the computational efficiency of the circuit during the processing. Hence I have designed a specific Quantum Error Detection model using the repetition code and syndrome measurement technique. This model detects the type as well as the position of the error inside a circuit with 100 % precision in Simulator & 48% - 50% precision in physical hardware.

Part 2: deals with an optimization of the circuit. As most of the Quantum Gates requires large data storage and RAM, it was part of my project to find the most efficient and optimized circuit which requires less cost to perform Quantum error correction operation. In this part of my project I got to exploit on many optimization techniques used precisely for circuit gate optimization.

Entire project was implemented in Jupyter notebook, using Py3 version, and Qiskit SDK with some added libraries {Aer, BasicAer, Ignis, IBMQ provider}for building gates and graphs to showcase the results and observations.

And the source link for my project can be found in this GitHub repo [10].

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

ABOUT THE ORGANISATION

CHAPTER 1

ABOUT THE ORGANISATION

1.1 Company Overview

QWorld (Association) is a non-profit global organization that brings quantum computing researchers & enthusiasts together. Our main goal is to popularize quantum technologies and software. Also, through education and skill development opportunities, QWorld is training the next generation of quantum scientists.

1.2 Corporate Philosophy

1.2.1 VISION

Our main goal is to popularize quantum technologies and software. We are looking for enthusiastic individuals, groups, institutions, organizations, and companies to work and operate together to develop open quantum ecosystem.

1.2.2 MISSION

QWorld is a global network of individuals, groups, and communities collaborating on education and implementation of quantum technologies and research activities.

1.2.3 OBJECTIVE

QWorld has International collaborations by establishing quantum groups known as QCousin's. These groups organize events in their regions, often communicating in the native language. QResearch's goal is to bring together researchers to foster collaboration and match students with mentors to develop a short-term research project. QWorld is also committed to increasing women's representation in the STEM field, and for this reason, QWomen was created. Finally, the QJunior program is focused on high school students.

1.3 Solutions

1.3.1 Industrial Solutions

QWorld provides a platform for many industries for discreet applications such as:

[1] Design:

Many products are designed and pre-tested using computer simulation. Automotive and aerospace hardware components and subcomponents are 3D-modeled with individual engineering safety margins. These margins can accumulate, culminating in products that are over-engineered, overweight, or higher cost than necessary, which can stifle their commercial viability.

[2] Control:

Modern control processes in manufacturing test the limits of advanced analytics, especially when employing machine learning and analysing multiple variables.

Quantum computing might help find new correlations in data, enhance pattern recognition, and advance classification beyond the capabilities of classical computing. The combination of quantum computing and machine learning, as well as its application to optimization,

[3] Supply:

Supply chains are shifting from a linear model with discrete, sequential, event-driven processes to a more responsive organic model based on evolving real-time market demands and up-to-the-minute availability of key components. Adding to the digital supply chain toolbox of Industry 4.0, quantum computing potentially could accelerate decision-making and enhance risk management to lower operational costs, as well as reduce lost sales because of out-of-stock or discontinued products. Enhancing competitive agility, quantum computing might completely transform the supply chain over time, adaptively redesigning it to optimize vendor orders and accompanying logistics using dynamic near-real-time decision-making based on changing market demands.

CHAPTER 2

ABOUT THE DEPARTMENT

2.1 QResearch

QResearch is a department of the QAcademy which aims to popularize and, what is most important, create such results. We encourage quantum researchers and enthusiasts to share their knowledge and expertise on the most current results in quantum technologies and computing.

Our Team

Coordinators: Adoam Glos (QPoland) and Zoltán Zimborás (QHungary)

Vice-coordinator: Arita Sarkar

Members: Krista Petersone (QLatvia), Engin Bac (Qturkey), Aleksandra Lipińska (QPoland), Abu Sayyed Md. Nasif, Tasir Olmez.

Under this department many distinguished activities are organised for the upcoming quantum researchers and enthusiasts. Some of them are:

[1] Research projects

We believe that high-quality research requires a simple reconstruction of results and easy access to the results by anyone. Finally, sharing our experiences is important – the more people with different skills join, the more likely it is that the research outcomes will meet high-quality criteria.

Therefore we initiate a Research project program.

[2] **Study Groups**

An excellent way of understanding a large piece of material is to meet in a small group and simply go through it together. This is the purpose of study groups: A bunch of people take the same material, split the content among themselves, and share their thoughts. A study group is a virtual place where you can study together with your teammates' chosen materials.

[3] **QIntern program**

QIntern is an internship program that allows collaborative work between more experienced people and those willing to learn more. Projects led by quantum researchers and enthusiasts focus on creating new software, educational materials or research results. However, what is even more important is that the participants have a chance to meet new people, which may result in longer collaborations.

During the QIntern 2020 pilot program a dozen of projects, most of which ended successfully. Similarly even QIntern 2021, where over 25 projects with almost two hundred interns and it continues. In the last days of the event, participants had an opportunity to share their outcomes and present their new educational materials, software, or research outcomes.

[4] **QHackatons**

QHackatons are a perfect way for intense knowledge and skills absorption. In 2019 QTurkey organized the first Quantum Programming Hackathon in Ankara, Turkey. Soon, October 10-11, 2020, they will run the next edition of the event – Quantum Technology Hackathon.

CHAPTER 3

INTRODUCTION TO QUANTUM GATES

CHAPTER 3

INTRODUCTION TO QUANTUM GATES

3.1 Quantum Bits

A digital computer both stores and processes information using bits, which can be either 0 or 1. Physically, a bit can be anything that has two distinct configurations: one represented by "0", and the other represented by "1". It could be a light bulb that is on or off, a coin that is heads or tails, or any other system with two distinct and distinguishable possibilities. In modern computing and communications, bits are represented by the absence or presence of an electrical signal, encoding "0" and "1" respectively.

A quantum bit is any bit made out of a quantum system, like an electron or photon. Just like classical bits, a quantum bit must have two distinct states: one representing "0" and one representing "1". Unlike a classical bit, a quantum bit can also exist in superposition states, be subjected to incompatible measurements, and even be entangled with other quantum bits. Having the ability to harness the powers of superposition, interference and entanglement makes qubits fundamentally different and much more powerful than classical bits.

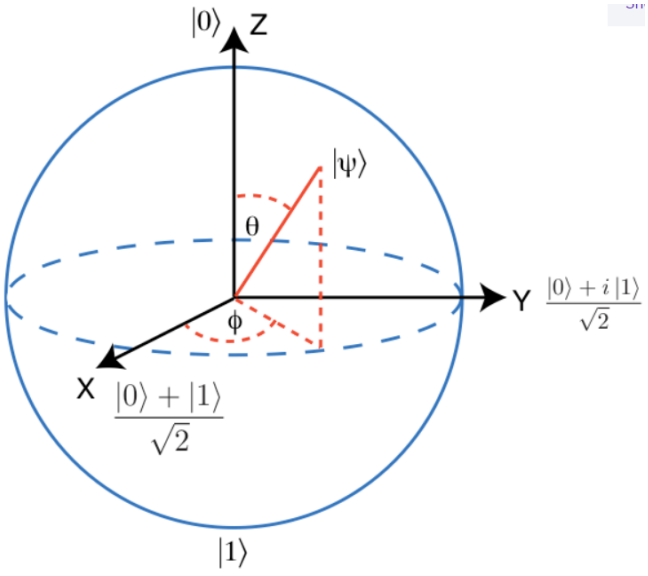


Figure 3.1: Bloch sphere Representation

The qubits or so called Quantum Bits could be represented by 2D vectors, and that their states are limited to the form:

$$|q\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle$$

Where θ and ϕ are real numbers. Using this equation the qubits are represented in a 2-unit sphere, where at the north and south poles lie two mutually.

When the qubit is measured, it collapses to one of the two poles. Which pole depends exactly on which direction the arrow in the Bloch representation as shown in Figure 3.1 points to: If the arrow is closer to the North Pole, there is larger probability to collapse to that pole; similarly for the South Pole. This introduces the notion of probability in the Bloch θ sphere: the angle of the arrow with the vertical axes corresponds to that probability. If the arrow happens to point exactly at the equator, there is 50-50 chance to collapse to any of the two poles.

3.2 The different Types of Gates implemented

In quantum computing and specifically the quantum circuit model of computation, a quantum logic gate (or simply quantum gate) is a basic quantum circuit operating on a small number of qubits. They are the building blocks of quantum circuits, like classical logic gates are for conventional digital circuits.

Unlike many classical logic gates, quantum logic gates are reversible. It is possible to perform classical computing using only reversible gates. Quantum gates are unitary operators, and are described as unitary matrices relative to some basis. Usually we use the computational basis, which unless we compare it with something, just means that for a d-level quantum system (such as a qubit, a quantum register, or qutrits and qudits. In this particular project only a selected quantum gates are utilized which are discussed in the further sections.

3.2.1 Pauli X gate:

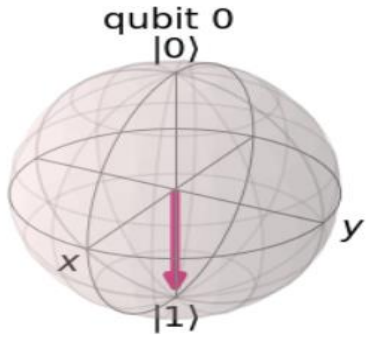
The Pauli-X gate is single-qubit rotation through π radians around the X-axis

The X-gate is represented by the Pauli-X matrix:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = |0\rangle\langle 1| + |1\rangle\langle 0|$$

The qubit's state vector is multiplied by the gate, which in turn effects the qubit. The X-gate switches the amplitudes of states.

Bloch Representation:



3.2.2 Pauli Y & Z Gate:

Similarly to the X-gate, the Y & Z gate is single-qubit rotation through π radians around the Y & Z axis.

Pauli matrices also act as the Y & Z-gates in the quantum circuits. They also respectively perform rotations by around the Y and Z-axis of the Bloch sphere.

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$Y = -i|0\rangle\langle 1| + i|1\rangle\langle 0| \qquad Z = |0\rangle\langle 0| - |1\rangle\langle 1|$$

3.2.3 Hadamard Gate [H Gate]

The Hadamard gate is a single-qubit operation that maps the basics state:

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

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

The Hadamard gate (H-gate) is a fundamental quantum gate. It allows to move away from the poles of the Bloch sphere and create a superposition of $|0\rangle$ and $|1\rangle$. It has the following matrix:

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

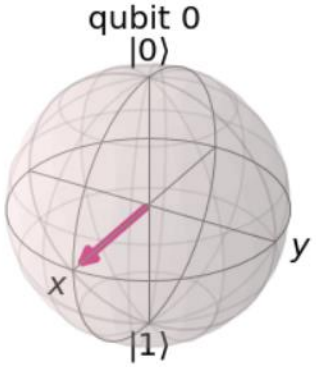
After the transformation:

$$H|0\rangle = |+\rangle$$

$$H|1\rangle = |-\rangle$$

This can be thought of as a rotation around the Bloch vector [1, 0, 1] (the line between the X & Y-axis), or as transforming the state of the qubit between the X and Z bases.

Bloch Representation:



3.2.4 The CNOT-Gate [CX-gate]:

The CNOT gate is two-qubit operation, where the first qubit is usually referred to as the control qubit and the second qubit as the target qubit. Expressed in basis states, the CNOT gate:

- Leaves the control qubit unchanged and performs a Pauli-X gate on the target qubit when the control qubit is in state |1\rangle|1⟩;
- Leaves the target qubit unchanged when the control qubit is in state |0\rangle|0⟩.

Table 4.1.4 shows the classical Not gate truth table:

Table 4.1.4: CNOT-Gate Truth Table

Input (t,c)	Output (t,c)
00	00
01	11
10	10
11	01

And acting on the 4D-statevector, it has one of the two matrices:

$$\mathrm{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \mathrm{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Depending on which qubit is the control and which is the target. Different books, simulators and papers order their qubits differently. In this case, the left matrix corresponds to the CNOT in the matrix above. This matrix swaps the amplitudes of |**01**> and |**11**> in the state vector.

$$|a\rangle = \begin{bmatrix} a_{00} \\ a_{01} \\ a_{10} \\ a_{11} \end{bmatrix}, \quad \text{CNOT}|a\rangle = \begin{bmatrix} a_{00} \\ a_{11} \\ a_{10} \\ a_{01} \end{bmatrix} \begin{matrix} \\ \leftarrow \\ \\ \leftarrow \end{matrix}$$

3.2.5 Introduction to Quantum Error Correction

Quantum error correction (QEC) is used in quantum computing to protect quantum information from errors due to decoherence and other quantum noise. Quantum error correction is theorised as essential to achieve fault-tolerant quantum computation that can reduce the effects of noise on stored quantum information, faulty quantum gates, faulty quantum preparation, and faulty measurements.

Classical error correction employs redundancy. The simplest albeit inefficient approach is the repetition code. The idea is to store the information multiple times, and—if these copies are later found to disagree—take a majority vote; e.g. suppose we copy a bit in the one state three times. Suppose further that a noisy error corrupts the three-bit state so that one of the copied bits is equal to zero but the other two are equal to one. Assuming that noisy errors are independent and occur with some sufficiently low probability p, it is most likely that the error is a single-bit error and the transmitted message is three ones. It is possible that a double-bit error occurs and the transmitted message is equal to three zeros, but this outcome is less likely than the above outcome. In this example, the logical information was a single bit in the one state, the physical information are the three copied bits, and determining what logical state is encoded in the physical state is called decoding. Similar to classical error correction, QEC codes do not always correctly decode logical qubits, but their use reduces the effect of noise.

Copying quantum information is not possible due to the no-cloning theorem. This theorem seems to present an obstacle to formulating a theory of quantum error correction. But it is possible to spread the (logical) information of one qubit onto a highly entangled state of several (physical) qubits. Peter Shor first discovered this method of formulating a quantum error correcting code by storing the information of one qubit onto a highly entangled state of nine qubits.

Classical error correcting codes use a syndrome measurement to diagnose which error corrupts an encoded state. An error can then be reversed by applying a corrective operation based on the syndrome. Quantum error correction also employs syndrome measurements. It performs a multi-qubit measurement that does not disturb the quantum information in the encoded state but retrieves information about the error. Depending on the QEC code used, syndrome measurement can determine the occurrence, location and type of errors. In most QEC codes, the type of error is either a bit flip, or a sign (of the phase) flip, or both (corresponding to the Pauli matrices X, Z, and Y). The measurement of the syndrome has the projective effect of a quantum measurement, so even if the error due to the noise was arbitrary, it can be expressed as a combination of basis operations called the error basis (which is given by the Pauli matrices and the identity). To correct the error, the Pauli operator corresponding to the type of error is used on the corrupted qubit to revert the effect of the error.

The syndrome measurement provides information about the error that has happened, but not about the information that is stored in the logical qubit—as otherwise the measurement would destroy any quantum superposition of this logical qubit with other qubits in the quantum computer, which would prevent it from being used to convey quantum information.

3.3 The Problem of Errors

Errors occur when some spurious operation acts on the qubits. Their effects cause things to go wrong in the circuits. The strange results one may have seen when running on real devices is all due to these errors.

There are many spurious operations that can occur, but it turns out that it is assumed that there are only two types of error: bit flips and phase flips.

Bit flips have the same effect as the **X** gate. They flip the **|0⟩** state of a single qubit to **|1⟩** and vice-versa. Phase flips have the same effect as the z gate, introducing a phase of **−1** into superposition. Put simply, they flip the **|+⟩** state of a single qubit to **|−⟩** and vice-versa.

The reason it can be thought of any error in terms of just these two is because any error can be represented by some matrix, and any matrix can be written in terms of the matrices **X** and **Z**. Specifically, for any single qubit matrix **M**,

$$M = \alpha I + \beta X + \gamma XZ + \delta Z,$$

For some suitably chosen values α, β, γ and δ

So whenever this matrix is applied to some single qubit state |ψ⟩ one will get,

$$M|\psi\rangle = \alpha|\psi\rangle + \beta X|\psi\rangle + \gamma XZ|\psi\rangle + \delta Z|\psi\rangle$$

The resulting superposition is composed of the original state, the state one would have if the error was just a bit flip, the state for just a phase flip and the state for both. If one had some way to measure whether a bit or phase flip happened, the state would then collapse to just one possibility. And the complex error would become just a simple bit or phase flip.

Quantum error detection is all about the how to detect whether it is a bit flip or phase flip (or both). Hence the rest of the section is about introducing the errors in the circuit and detecting.

3.4 A Concise Entangled Circuit Implementation

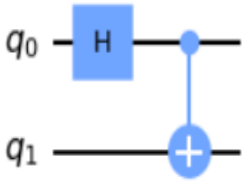
Generally in quantum computation initially the quantum circuits must be in entanglement. Hence, a pair of entangled qubits is created.

```
from qiskit import QuantumCircuit, Aer

# Make an entangled pair
qc_init = QuantumCircuit(2)
qc_init.h(0)
qc_init.cx(0,1)

# Draw the circuit
display(qc_init.draw('mpl'))

# Get an output
qc = qc_init.copy()
qc.measure_all()
job = Aer.get_backend('qasm_simulator').run(qc)
job.result().get_counts()
```




```
{'00': 498, '11': 526}
```

Figure 3.4: {Code}: Creating Entanglement State

Since, the qubits are now in “Bells State” (entangled pair). The Figure 3.4, shows the expected results **00** and **11** which are occurring with equal probability [**498**] and [**526**]. In further sections the two types of possible error are introduced.

3.5 Bit Flip 'Error'

A bit flip error is specific type of error where the qubits computational state flips from **1** to **0** or vice versa. However a bit flip can be corrected using the bit flip code. This is a 3 qubit circuit that makes use of 2 ancillary qubits to correct 1 qubit.

A bit flip occurs when copying the data and one of the bits changes so that it's incorrect. A value of 1 incorrectly becomes a zero, or vice versa. Bit flips that lead to bug checks are a common way that Windows detects a hardware problem (e.g., bad memory, an overheating CPU).

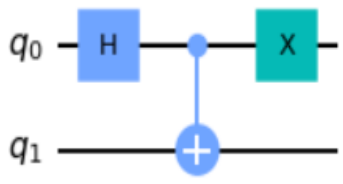
But what happens when one have the same circuit, but with a bit flip 'error' inserted manually.

```
# Make bit flip error
qc_insert = QuantumCircuit(2)
qc_insert.x(0)

# Add it to our original circuit
qc = qc_init.copy()
qc = qc.compose(qc_insert)

# Draw the circuit
display(qc.draw('mpl'))

# Get an output
qc.measure_all()
job = Aer.get_backend('qasm_simulator').run(qc)
job.result().get_counts()
```




```
{'01': 521, '10': 503}
```

Figure 3.5.1: {Code}: Creating Bit Flip 'Error'

Now the results are different: **01** and **10** as seen in the Figure 3.5.1. The two bit values have gone from always agreeing to always disagreeing. In this way, the effect of the error is detected.

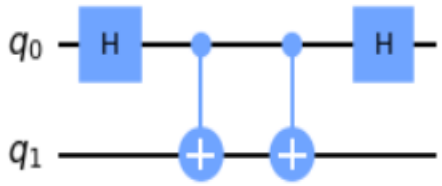
Another way to detect is to undo the entanglement with a few more gates.

```
# Undo entanglement
qc_syn = QuantumCircuit(2)
qc_syn.cx(0,1)
qc_syn.h(0)

# Add this after the error
qc = qc_init.copy()
qc = qc.compose(qc_syn)

# Draw the circuit
display(qc.draw('mpl'))

# Get an output
qc.measure_all()
job = Aer.get_backend('qasm_simulator').run(qc)
job.result().get_counts()
```




```
{'00': 1024}
```

Figure 3.5.2: {Code}: Detection of Bit Flip 'Error'

If there are no errors, we return to the initial **|00⟩** state.

As seen in Figure 3.5.2, a circuit with all the components which are introduced so far: the initialization **qc_init,** the inserted error in **qc_insert** and the final **qc_syn** which ensures that the final measurement gives a nice definite answer.

3.6 Phase Flip 'Error'

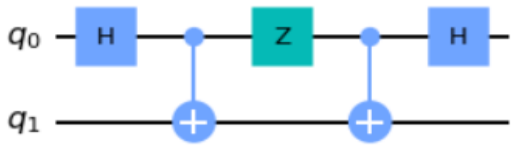
```
# Define an error
qc_insert = QuantumCircuit(2)
qc_insert.z(0)

# Undo entanglement
qc_syn = QuantumCircuit(2)
qc_syn.cx(0,1)
qc_syn.h(0)

# Add this after the error
qc = qc_init.copy()
qc = qc.compose(qc_insert)
qc = qc.compose(qc_syn)

# Draw the circuit
display(qc.draw('mpl'))

# Get an output
qc.measure_all()
job = Aer.get_backend('qasm_simulator').run(qc)
job.result().get_counts()
```




```
{'01': 1024}
```

Figure 3.6.1: {Code}: Creating a Phase Flip 'Error'

The Figure 3.6.1 shows exactly how the errors are attained. Both the bit and phase flips can be detected. The bit value on the left is 1 only if there is a bit flip (*and so if we have inserted an* ***x(0)*** *or* ***x(1)).*** The bit on the right similarly implicates there is a phase flip (*an inserted* ***z (0)*** *or* ***z (1)).*** This ability to detect and distinguish bit and phase flips is very useful. But it is not quite useful enough. It is only predict ***what type*** of errors are occurring, but not ***its place***. Without more detail, it is impossible to figure out how to remove the effects of these operations from the computations. For quantum error detection there is therefore a need for something better.

Figure 3.6.2: **The Expected Result {Quantum State}:**

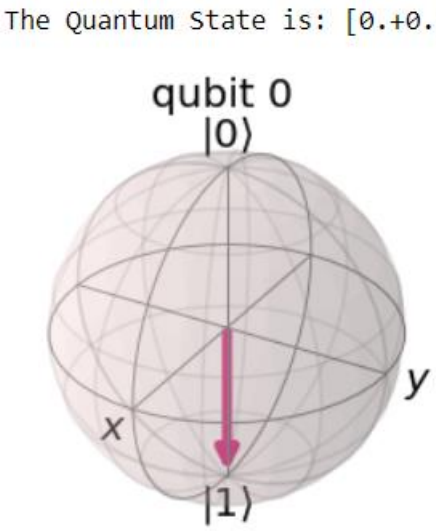
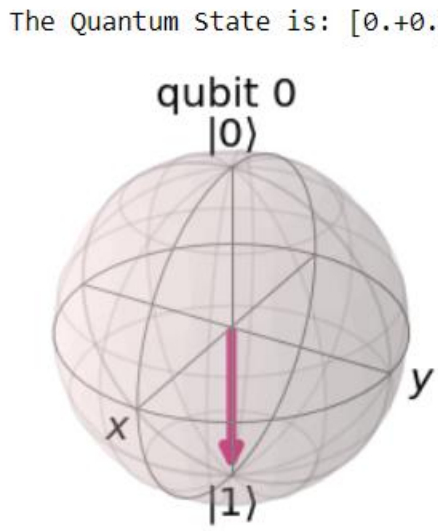


Figure 3.6.3: **The Obtained Result {With Error Induced}:**



The Figure 3.6.3 shows that, only the phase has been flipped but not the Bit. As seen in Figure 3.6.2 the state has remained at pointing towards **|1⟩** even after the circuit changed**.**

3.7 QISKIT Software Development Kit (SDK)

Qiskit is an open-source software development kit (SDK) for working with quantum computers at the level of circuits, pulses, and algorithms. It provides tools for creating and manipulating quantum programs and running them on prototype quantum devices on IBM Quantum Experience or on simulators on a local computer.

It follows the circuit model for universal quantum computation, and can be used for any quantum hardware (currently supports superconducting qubits and trapped ions) that follows this model.

Qiskit was founded by IBM Research to allow software development for their cloud quantum computing service, IBM Quantum Experience. Contributions are also made by external supporters, typically from academic institutions.

The primary version of Qiskit uses the Python programming language. Versions for Swift and JavaScript were initially explored, though the development for these versions have halted. Instead, a minimal re-implementation of basic features is available as MicroQiskit, which is made to be easy to port to alternative platforms.

A range of Jupyter notebooks are provided with examples of quantum computing being used. Examples include the source code behind scientific studies that use Qiskit, as well as set of exercises to help people to learn the basics of quantum programming. An open source textbook based on Qiskit is available as a university level Quantum algorithms or quantum computation course supplement.

3.7.1 Tools Used

Qiskit is made up of elements that work together to enable quantum computing. The central goal of Qiskit is to build a software stack that makes it easy for anyone to use quantum computers, regardless of their skill level or area of interest; Qiskit allows users to easily design experiments and applications and run them on real quantum computers and/or classical simulators. Qiskit provides the ability to develop quantum software both at the machine code level of OpenQASM, and at abstract levels suitable for end-users without quantum computing expertise.

This functionality of the programming is provided by the following distinct libraries.

a. ***Qiskit Terra*** :

The element Terra is the foundation on which the rest of Qiskit is built. Qiskit Terra provides tools to create quantum circuits at or close to the level of quantum machine code.

b. Qiskit Aer:

The element Aer provides high-performance quantum computing simulators with realistic noise models. In the near-term, development of quantum software will depend largely on simulation of small quantum devices.

c. Qiskit Ignis:

The element Ignis provides tools for quantum hardware verification, noise characterization, and error correction. Ignis is a component that contains tools for characterizing noise in near-term devices, as well as allowing computations to be performed in the presence of noise.

CHAPTER 4

TASKS PERFORMED

CHAPTER 4

TASKS PERFORMED

Goal: Create circuits which can detect `x` and `z` errors on two qubits, also with the lowest possible Cost Estimation.

4.1 Implement a Cost Efficient Circuit for Error Detection

4.1.1 Mapping the Circuit using *STABILIZER OPERATOR* on 3 Qubits:

The qubits are to be laid out as follows, with the code qubits forming the corners of four triangles, and the syndrome qubits living inside each triangle.

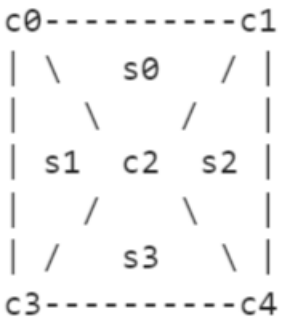


Figure 4.1.1: Circuit Layout Graph

According to Figure 4.1.1, for each triangle a stabilizer operation is associated on its three qubits. For the qubits on the sides, the stabilizers are **ZZZ**. For the top and bottom ones, they are **XXX**.

The syndrome measurement circuit corresponds to a measurement of these observables. This is done in a similar way to surface code stabilizers *(in fact, this code is a small version of a surface code).*

4.1.2 Creating the *SYNDROME* Circuit

The syndrome circuit is modified to make sure all CNOT gates are between connected qubits. This modified syndrome circuit is coded as shown in Figure 4.1.2.1 and generated circuit is seen in Figure: 4.1.2.2

```
qc_syn = QuantumCircuit(code,syn,out)


# Left ZZZ
qc_syn.cx(code[0],syn[1])
qc_syn.cx(code[2],syn[1])
qc_syn.cx(code[3],syn[1])
qc_syn.barrier()

# Right ZZZ
qc_syn.cx(code[1],syn[2])
qc_syn.cx(code[2],syn[2])
qc_syn.cx(code[4],syn[2])
qc_syn.barrier()

# Top XXX
qc_syn.h(syn[0])
qc_syn.cx(syn[0],code[0])
qc_syn.cx(syn[0],code[1])
qc_syn.cx(syn[0],code[2])
qc_syn.h(syn[0])
qc_syn.barrier()

# Bottom XXX
qc_syn.h(syn[3])
qc_syn.cx(syn[3],code[2])
qc_syn.cx(syn[3],code[3])
qc_syn.cx(syn[3],code[4])
qc_syn.h(syn[3])
qc_syn.barrier()


# Measure the auxilliary qubits
qc_syn.measure(syn,out)
qc_syn.draw('mpl')
```

Figure 4.1.2.1: {Code}: Creating a Syndrome Circuit

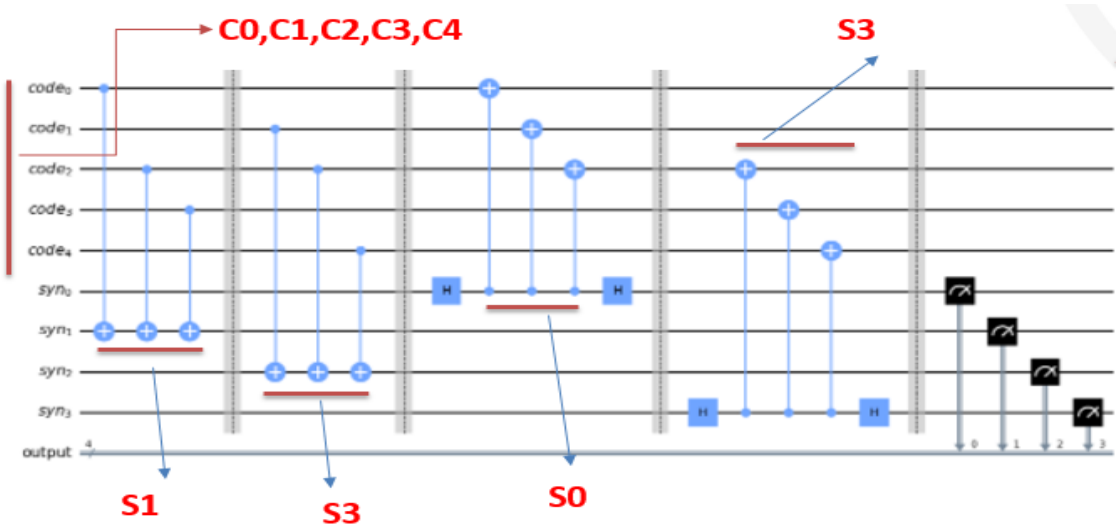


Figure 4.1.2.2: {Circuit}: Syndrome Circuit

4.1.3 Connectivity chart for analyzing the Syndrome Circuit

```
c0....s0....c1
:      :     :
:      :     :
s1....c2....s2
:      :     :
:      :     :
c3....s3....c4
```

4.1.4 Mapping Logical Qubits from circuit to Hardware

// initializing the Array to ***'initial_layout '***

```
initial_layout = [0,2,6,10,12,1,5,7,11]
```

The above snippet code implies that **code[0]** will map to qubit 0 on the device, **code[1]** to qubit 2, and so on. After looking at this and checking with qubits have a direct connection on the device, we see that the only **CNOT** that will not work on hardware is the first one on the Right **ZZZ** block, i.e. between **code[1]** and **syn[2].**

The easiest thing to do here is to build a series of gates that perform the same operation than a **CNOT** without the two gates being actually connected to each other. And doing this is easy using an **ancilla** qubit. At this point in the circuit, there are two options for **ancilla** qubits, **syn [0]** and **syn [3];** syn **[1]** cannot be used since it has already been used by the first block and it's best to not change its state. 1 chose to use **syn[0].** The code is rather easy. First, a **CNOT** gate is applied with **code [1]** as control and **syn [0].** Since **syn[0]** is still in the **0** state, it will take on the state of code[1] after the **CNOT**. Then, a CNOT gate is applied with **syn [0]** as control and **syn [2]** as target, this will have the same effect as a **CNOT** between code [1] and **syn [2]** directly. Finally, the first **CNOT** is applied again to reverse **syn [0]** to the **0** state.

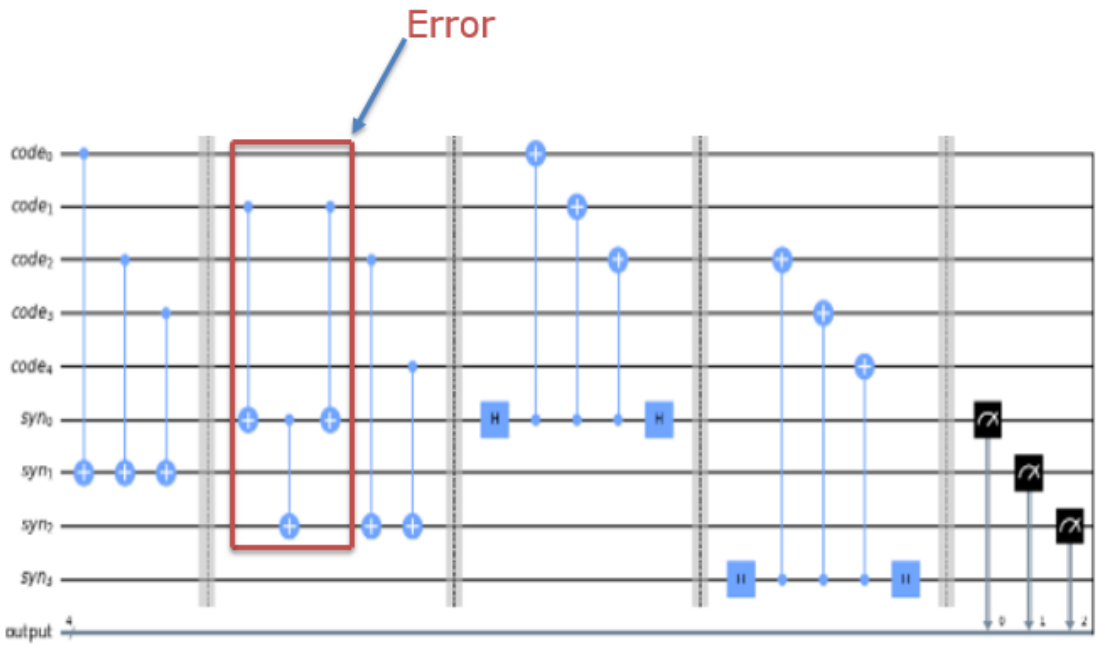


Figure 4.1.4: Circuit after commits *{with respect to Layout}*

4.1.5 Fake Tokyo's Architecture

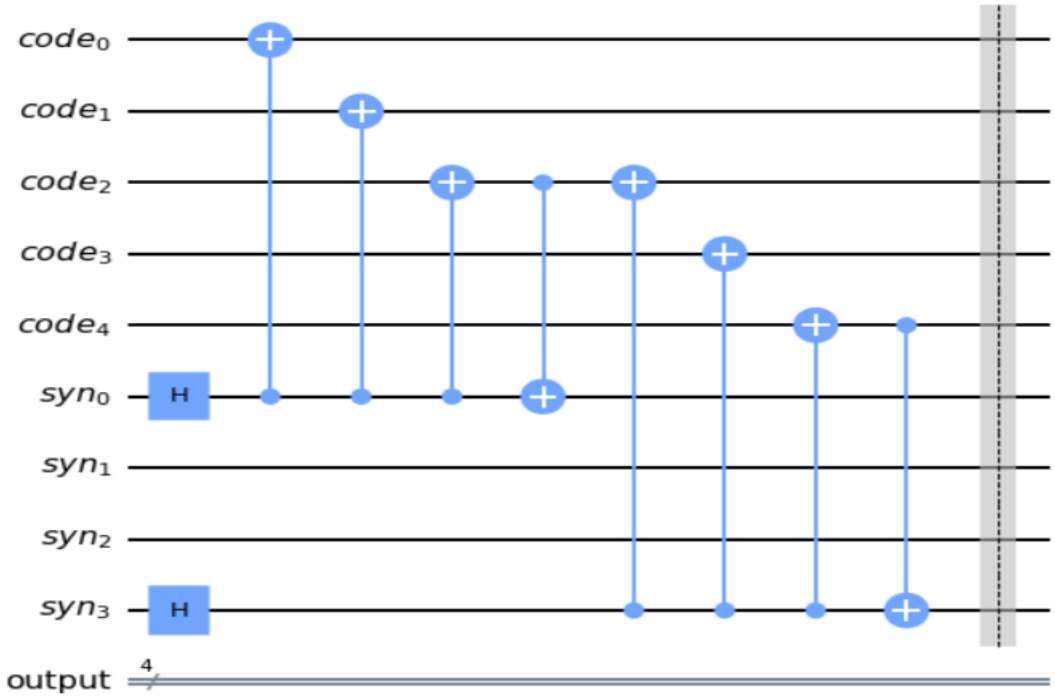


Figure 4.1.5: {Circuit}: Creating a Fake-Tokyo Architecture

Figure 4.1.4 depicts about the error circuit which is artificially appended to scrutinize the efficiency of the Error Model.

Figure 4.1.5 shows the generated circuit which is used for mapping the hardware with the simulator in order to observe the changes occurred once the error was introduced.

4.1.6 Inducing the *'Error'* to Circuit Model:

To check if the circuit runs correctly, the error qubits are defined and some artificial errors are inserted and the code snippet is shown in Figure below to make sure one can detect them:

```
error_qubits = [0,4]

def insert(errors,error_qubits,code,syn,out):

    qc_insert = QuantumCircuit(code,syn,out)

    if 'x0' in errors:
        qc_insert.x(error_qubits[0])
    if 'x1' in errors:
        qc_insert.x(error_qubits[1])
    if 'z0' in errors:
        qc_insert.z(error_qubits[0])
    if 'z1' in errors:
        qc_insert.z(error_qubits[1])

    return qc_insert

for error in ['x0','x1','z0','z1']:

    qc = qc_init.compose(insert([error],error_qubits,code,syn,out)).compose(qc_syr
    job = Aer.get_backend('qasm_simulator').run(qc)

    print('\nFor error '+error+':')
    counts = job.result().get_counts()
    for output in counts:
        print('Output was',output,'for',counts[output],'shots.')
```

4.1.7 Detection of both {Error Places & Types}

```
For error x0:
Output was 0010 for 1024 shots.

For error x1:
Output was 0100 for 1024 shots.

For error z0:
Output was 0001 for 1024 shots.

For error z1:
Output was 1000 for 1024 shots.
```

Error Format:

Z1	X1	X0	Z0

Here we see each Qubit in the output is 1:

1. 0010 → { X0 } → Bit Flip Error
2. 0100 → { X1 } → Bit Flip Error
3. 0001 → { Z0 } → Phase Flip Error
4. 1000 → { Z1 } → Phase Flip Error

Figure 4.1.6: Error Location and Type

In the error format **[Z1, X1, X0, Z0]** as seen in the Figure 4.1.6, **[Z]** denotes Phase flip error and **[X]** denotes bit flip error. As per the inference obtained if there is in $|1\rangle$ the place of $|0\rangle$ it indicates the presence of an error. With reference to the error format all the four types of errors with its respective locations are precisely detected.

4.1.8 Expected output

The initialization circuit prepares an eigenstate of these observables, such that the output of the syndrome measurement will be ***0000*** with certainty.

Hence, the location and type of Error is detected and displayed with **100%** precision or accuracy.

4.2 Estimating the cost of Circuit

```
from qc_grader import grade_ex3
grade_ex3(qc_init,qc_syn,error_qubits,initial_layout)
```

```
Grading your answer for ex3. Please wait...

Congratulations 🎉! Your answer is correct.
Your cost is 226.
```

To calculate the cost:

*a. CX-Gate {10} : 22 * 10 = 220*

*b. H-Gate {1} : 6 * 1 = 6*

Total: 226

In order to calculate the cost of a circuit, the amount of quantum gates are to be calculated. Hence, as seen the Figure 4.2 there are about **22** CX-gate and **6** H-gate in total.

By default the gates are initialized with a cost as below:

[1] CX-gate: **10**

[2] H-gate: **6**

From the above tallying, the total effective cost the obtained circuit is **226.** At the moment, it is estimated that **226** was the best possible optimized cost we could obtain.

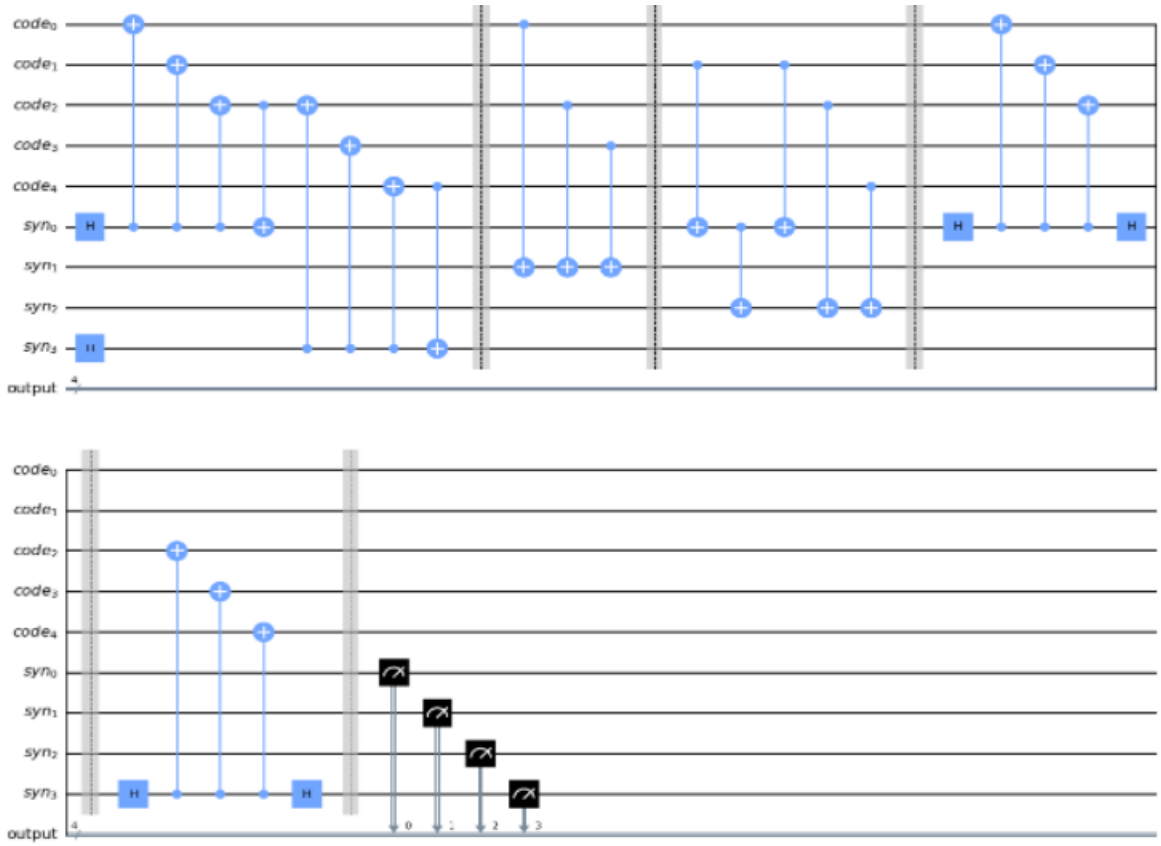


Figure 4.2: {Circuit}: Cost Effective Circuit

CHAPTER 5

REFLECTION NOTES

CHAPTER 5

CONCLUSION

5.1 Reflection Notes

During the period of internship at QIntern-21 organized by QWorld Association, I had a great exposure to Qiskit SDK which they primarily use for designing Quantum application development. My task involved designing the Error Detection Model which is Cost Effective. The design code were written in Python code and simulated on IBMQ. Going through all these procedures I learnt to perform verification Visualization graphs, emulator error model, and functional error due to gate errors.

I got valuable insight into the field of Quantum error Correction & Detection from Industry Professionals and also the different career opportunities for a fresher in the field of Quantum Application developer and also the future growth prospects in this field.

\

5.2 References

[1] R. Landauer, "Is quantum mechanics useful?" Phil. Trans. Roy. Soc. London. Series A, Phys. Eng. Sci., vol. 353, no. 1703, pp.367-376, 1995, doi:10.1098/rsta.1995.0106.

[2] I. L. Chuang, R. Laflamme, P. W. Shor, and W. H. Zurek, "Quantum computers, factoring, and decoherence," Science, vol. 270, no. 5242, pp. 1633–1635, 1995, doi: 10.1126/science.270.5242.1633.

[3] W. G. Unruh, "Maintaining coherence in quantum computers," Phys. Rev. A, vol. 51, no. 2, 1995, Art. No. 992, doi: 10.1103/PhysRevA.51.992.

[4] G. M. Palma, K.-A. Suominen, and A. Ekert, "Quantum computers and dissipation," Proc. Roy. Soc. London. Series A, Math., Phys. Eng. Sci., vol. 452, no. 1946, pp. 567–584, 1996, doi: 10.1098/rspa.1996.0029.

[5] D. A. Lidar and T. A. Brun, Quantum Error Correction. Cambridge, U.K.: Cambridge Univ. Press, 2013, doi: 10.1017/CBO9781139034807.

[6] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, "Topological quantum memory," J. Math. Phys., vol. 43, no. 9, pp. 4452–4505, 2002.

[7] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," Phys. Rev. A, vol. 52, no. 4, 1995, Art. No. R2493, doi: 10.1103/physreva.52.r2493.

[8] Introduction to Quantum Error Correction: https://qiskit.org/textbook/ch-quantum-hardware/error-correction-repetition-code.html.

[9] Project Source code: https://github.com/ShisheerKaushik24/IBM-Quantum-challenge-2021-/blob/master/ex3.ipynb.

[10] Internship website {Project 17: Mentor: Zeki Seskir}:https://qworld.net/qintern-2021

[11] Qiskit SDK: https://qiskit.org/documentation/getting_started.html.