CS 595 – Termination Project,
Binghamton University

Project Report

Title: Quantum Error Correction with Shor Code

Under the guidance of
Prof. Leslie C. Lander

Made by:
Name: Siddhesh Shinde
B-No: B00863074
Email: sshinde7@binghamton.edu

Image Source: IBM Quantum at CES 2020
https://newsroom.ibm.com/media-quantum-innovation

Table of Contents:

Introduction

Quantum computers have the potential to solve certain problems exponentially faster than classical computers. However, quantum systems are highly sensitive to noise and errors, which can cause the failure of quantum algorithms due to loss of quantum coherence. The threshold theorem states that there exists a certain threshold for error rates below which we can achieve truly fault-tolerant computation. With enough error correction, this threshold can be achieved. Noise comes in the form of heat, radiation, stray electromagnetic fields, mechanical vibrations and/or other environmental disruptors that would affect the quantum computer. Quantum error correction (QEC) is a crucial technique to protect quantum information from errors due to decoherence and other quantum noise. Error correcting codes are a way of encoding information in some way that protects it from some or all errors. Some of the most common are the Shor code, Bosonic, Toric code and the Surface code.

In this project, we explore Shor code, a QEC code to detect and correct errors in a quantum system. We implement the Shor code using a quantum simulator and analyze its performance using different types and rates of errors.

Scope

The scope of the project is to design smaller component circuits for encoding and correction of different types of errors. Then we implement the Shor code, simulate different error types and error rates and examine the errors that occur during execution of Shor code. Finally, we analyze Shor code and compare it with code without error correction.

Literature Survey

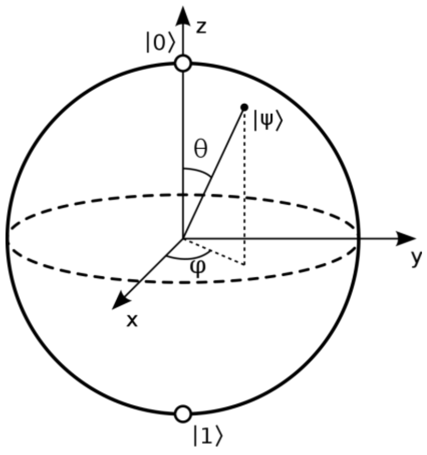
The project has been completed as a part of Qubit by Qubit's Introduction to Quantum Computing with IBM Quantum.

Technologies used:

IBM Quantum Lab: It's a cloud-based quantum computing service powered by Jupyter technology provided by IBM Quantum.
IBM Quantum Composer: A circuit composer with customizable set of tools to build, visualize and run quantum circuits on quantum hardware or simulators.
Qiskit: An open-source SDK for working with quantum computers at the level of pulses, circuits and application modules.

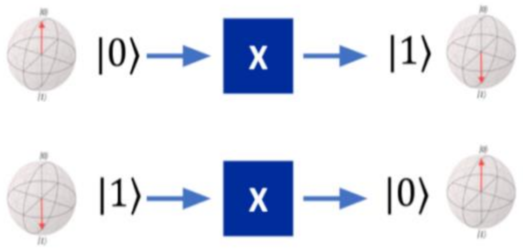
Quantum Computing Fundamentals:

Qubit: The basic unit of information in quantum computing is the qubit. It can exist in a superposition of its two "basis" states, which loosely means that it is in both states simultaneously.
Qubits can be represented using methods such as the ket notation or the Bloch sphere. The two basis states can be represented in ket notation as $|0\rangle$ and $|1\rangle$, and the bloch sphere in the figure, Any other state on bloch sphere represents a superpostion of $|0\rangle$ and $|1\rangle$. A superposition can be equal meaning that both basis states contribute equally to the state or unequally, meaning that either $|0\rangle$ contributes more or $|1\rangle$ does. If the state is closer to $|0\rangle$, it has a greater contribution from $|0\rangle$. And if it is closer to $|1\rangle$, it has a greater contribution from $|1\rangle$.

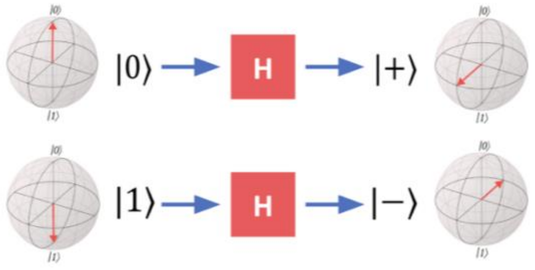


Quantum Gates and Circuits: Quantum gates manipulate or change the state of qubits. Superpostion, interference and entanglement can be created with gates. The operation of gates on qubits can be visualized as rotations on the Bloch sphere.

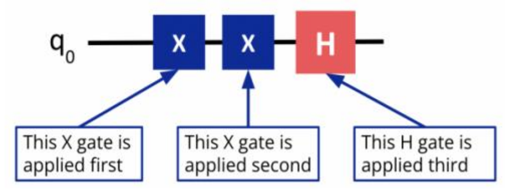
The X gate flips from $|0\rangle$ and $|1\rangle$ and vice versa.



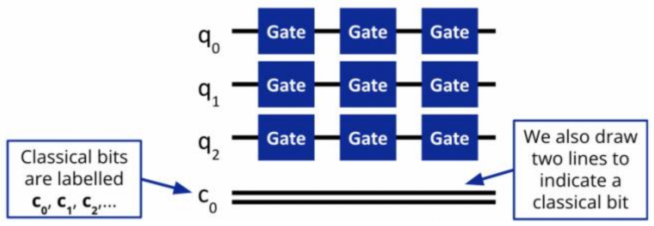
The H gate or Hadamard gate creates superposition states.



Quantum circuits are collections of quantum gates and in Qiskit, qubits always start as |0⟩. We can also have multiple qubits in our circuit or even classical bits.

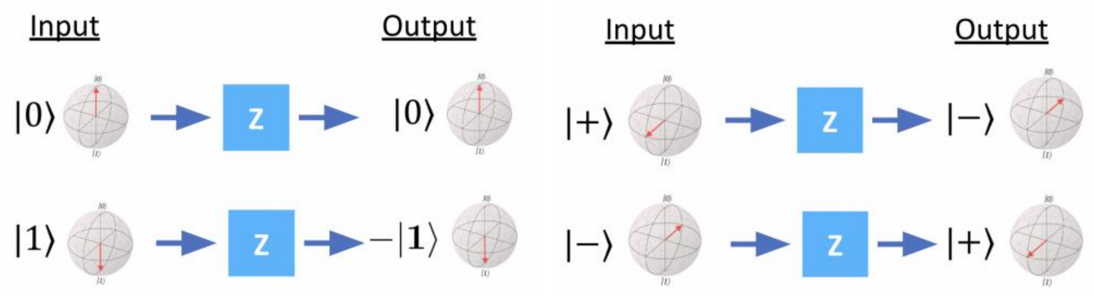


q_0
X
X
H
This X gate is applied first
This X gate is applied second
This H gate is applied third

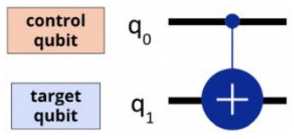


q_0
q_1
q_2
c_0
Gate
Classical bits are labelled $c_0, c_1, c_2, \ldots$
We also draw two lines to indicate a classical bit

The Z gate is a 180° rotation around the z-axis. The rotation is also called as a phase shift.

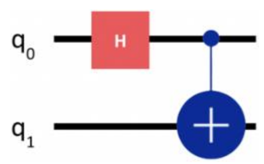


Input
Output
Input
Output
$|0\rangle$ Z $|0\rangle$
$|1\rangle$ Z $-|1\rangle$
$|+\rangle$ Z $|-\rangle$
$|-\rangle$ Z $|+\rangle$



The **Controlled X (CX or CNOT) gate** is a 2-qubit gate that follows these rules:

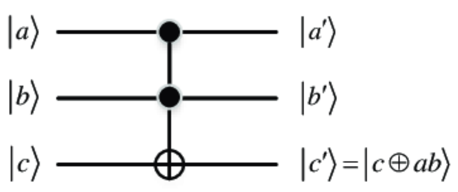
a. The control qubit never changes.
b. If the control qubit is 0, do nothing to the target qubit.
c. If the control qubit is not 0, apply an X gate to the target qubit.



Applying the CX gate to a superpostion creates entanglement.

The CCX or Toffoli Gate: This is a 3-qubit Controlled Controlled X gate with two control qubits that must be both 1 to flip the target.

Inputs			Ouputs		
a	b	c	a'	b'	c'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0



Methodology

Overview:

A Quantum Error Correction code can be broken down as a five step process:

1. Encoding: Logical states and gates are encoded into physical states and gates.
2. Sending over noisy channel: The physical state is exposed to noise and may incur errors.
3. Error Detection: Any errors in the state are detected and diagnosed using the error syndrome.
4. Error Correction: Any errors in the state are corrected.
5. Decoding: The final physical state is interpreted logically.

In this project, we implement, simulate and analyze the first ever error correction code proposed that can handle any single qubit errors (bit-flips, phase-flips, or others) using the Shor code. It is an extension of the bit-flip and phase-flip codes that avoid syndrome measurements where:

- A bit-flip error means $|0\rangle$ unintentionally turning into $|1\rangle$ or vice versa.
- A phase-flip error means $|+\rangle$ unintentionally turning into $|-\rangle$ or vice versa.

The code can be viewed as two levels of quantum error correction where:

1. The logical qubits are first encoded into a 3-qubit phase-flip code.
2. Each of the resulting physical qubits are encoded into a 3-qubit bit-flip code.

The project is broken into three parts:

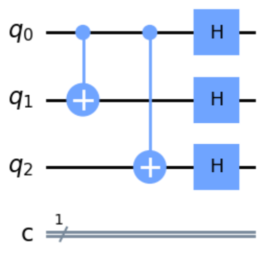
- Part 1: Defining the Components
    - Part 1.1: Phase-flip code
    - Part 1.2: Bit-flip code
- Part 2: Implementing the Shor code
- Part 3: Analyzing the Shor code

In this project, the errors were introduced in the circuit by applying quantum logic gates to simulate bit-flips and phase-flips errors. In Analysis of Shor code, we discuss the limitations of Shor code and compare its performance with code without error correction.

Implementation

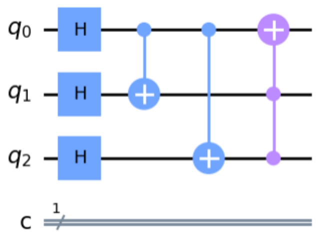
Defining the Components – Phase-flip code

Phase-flip encode: A 3-qubit phase-flip logical to physical encoding



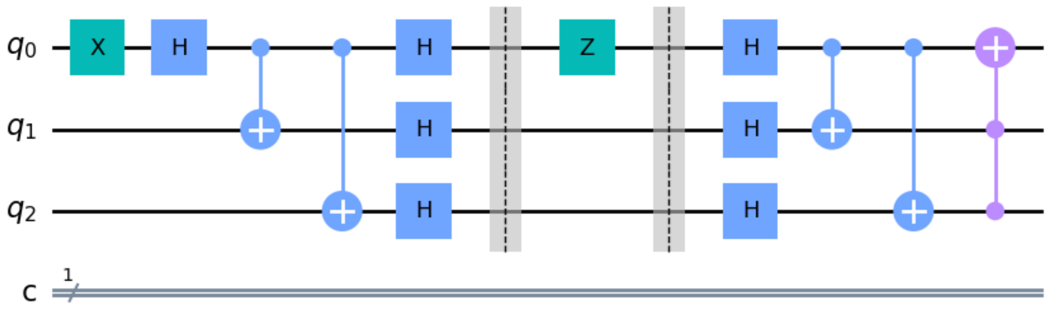
First, we initialize a circuit with 3 qubits. Then we entangle the state of main qubit with the other two qubits. Finally, we put all qubits in equal superpositions by applying H gates to correct |+⟩ and |-⟩ states from switching instead of |0⟩ and |1⟩.

Phase-flip correction: A 3-qubit phase-flip physical error detection and correction circuit



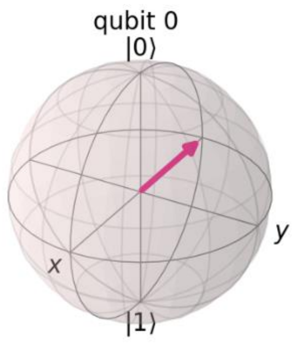
First, we initialize a circuit with 3 qubits. Then we detect any errors by checking that the phases of the 3 qubits match using H gates and CX gates. Then we correct any errors using a Toffoli gate.

Combining the encoding and correcting circuits and simulating the full phase-flip code for the |-⟩ state with a phase-flip error.

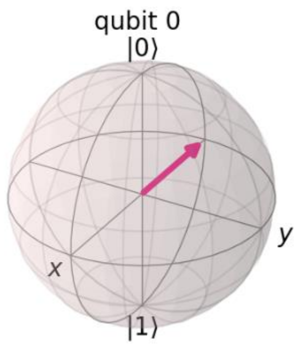


Comparing the initial and final quantum states (statevector).

```
initial_state.draw(output = 'bloch')
```



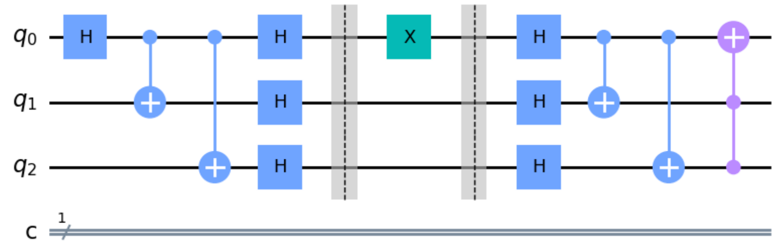
```
final_state.draw(output = 'bloch')
```



```
initial_state.equiv(final_state)
```

```
True
```

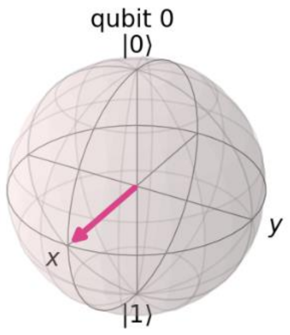
Simulating the full phase-flip code for the |+⟩ state with a bit-flip error on qubit 0 and comparing.



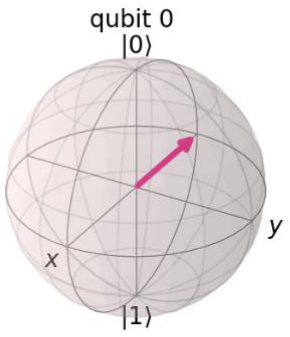
```
initial_state.equiv(final_state)
```

```
False
```

```
initial_state.draw(output = 'bloch')
```

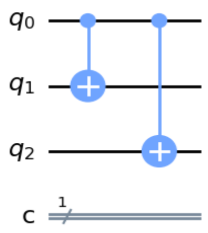


```
final_state.draw(output = 'bloch')
```

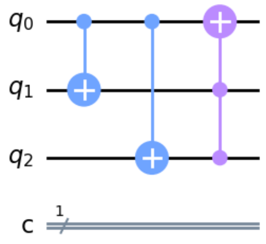


Defining the Components – Bit-flip code

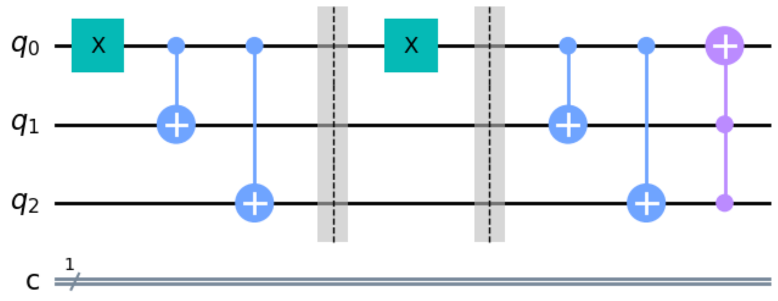
Bit-flip encode: A 3-qubit bit-flip logical to physical encoding



Bit-flip correction: A 3-qubit bit-flip physical error detection and correction circuit



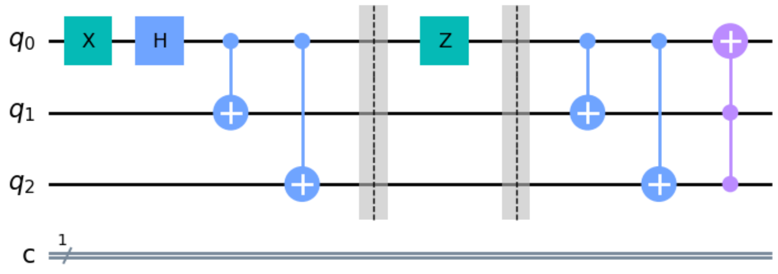
Combining the encode and correction circuits, simulating the full bit-flip code for the |1⟩ state with a bit-flip error on main qubit (qubit 0).



```
initial_state.equiv(final_state)
```

```
True
```

Simulating the full bit-flip code for the |-⟩ state with a phase-flip error on qubit 0.



```
initial_state.equiv(final_state)
```

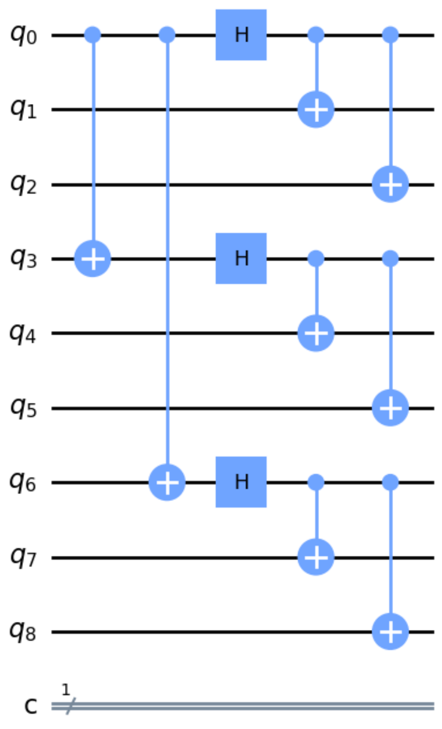
```
False
```

Implementing and Simulating Shor Code with errors:

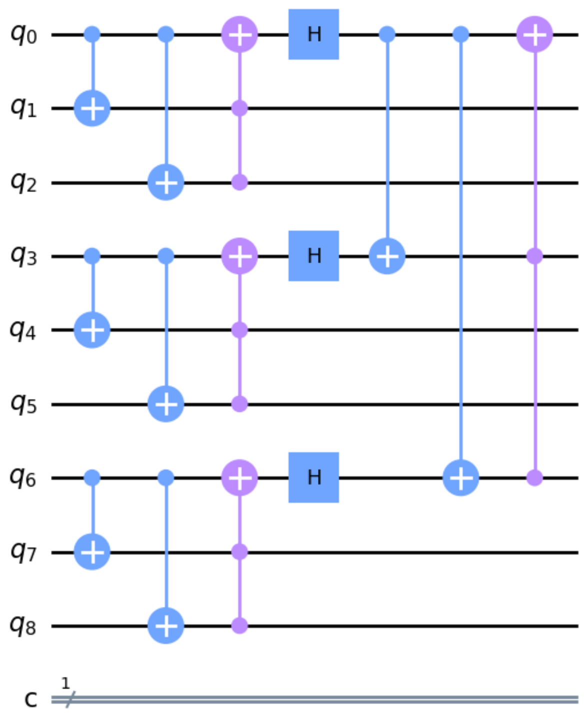
The Shor code is a QEC code that encodes one logical qubit in nine physical qubits and can correct for arbitrary errors in a single qubit. Eight out of the nine qubits are ancillary qubits that help detect and correct errors in one qubit, using repetition code that is configuration of quantum logic gates/circuits that get repeated in the final circuit.

- First, we initialize the Shor code's 9-qubits, that is 8 ancillary qubits and one main qubit.
- It works by taking the computational state of the main qubit and transferring it to the 3rd and 6th qubit which are used for correcting phase errors. This is done by applying CX gate with main qubit as controller and 3rd and 6th qubit as targets.
- Then these qubits are put into superposition using an H gate.
- Next, CX gates are applied with the main qubit as controller with 1st and 2nd as targets, 3rd as controller with 4th and 5th as targets and the 6th qubit as controller with 7th and 8th as targets.
- After this phase-flip or bit-flip errors can occur on the main qubit or other qubits. Then similar arrangement of CX gates from previous step is repeated.
- Toffoli gates are then applied to the main qubit, 3rd and the 6th qubit where the control qubits are the auxiliary qubits responsible for phase correction.
- After this step, H gate is applied to the main qubit, 3rd and 6th qubit.
- Then CX gate is applied where the main qubit is the controller and 3rd, and 6th qubits are targets.
- Finally, a Toffoli gate is applied with 3rd and 6th qubit as controllers and main qubit as target.

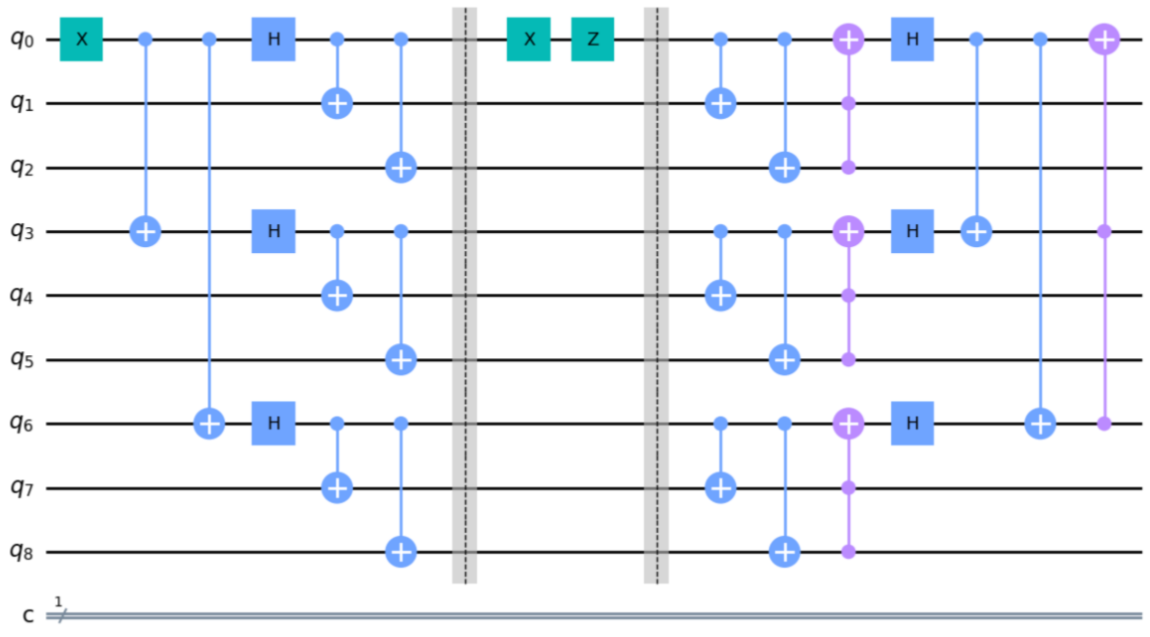
Shor Code Encoding



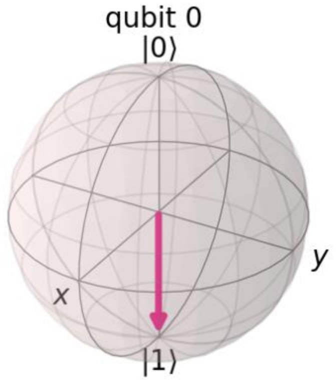
Shor Code Correction



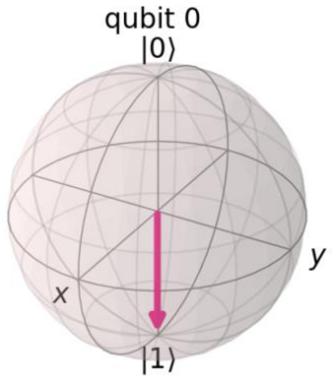
Combining Shor code's encode and correction circuits, simulating the full code for the |1⟩ state with bit-flip and phase-flip errors on qubit 0, and comparing the initial and final statevectors.



```
initial_state.draw(output = 'bloch')
```



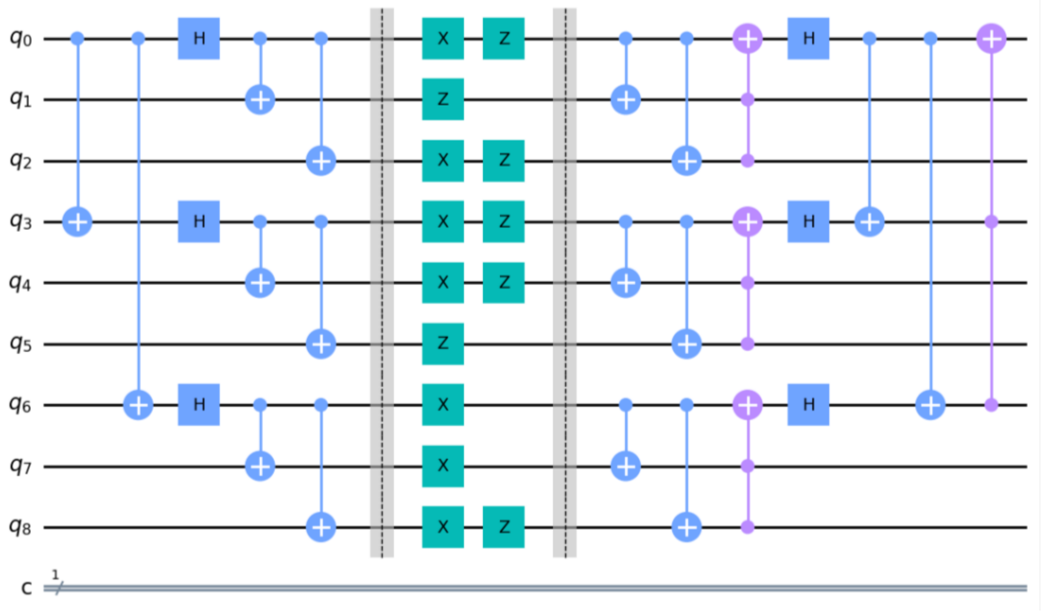
```
final_state.draw(output = 'bloch')
```



```
initial_state.equiv(final_state)
```

```
True
```

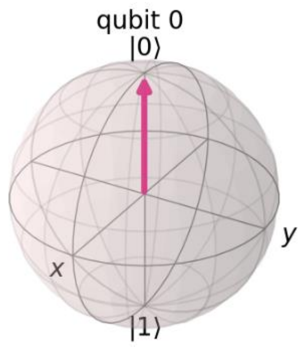
Simulating the Shor code with probability of bit and phase-flip errors occurring as 70% of the time.



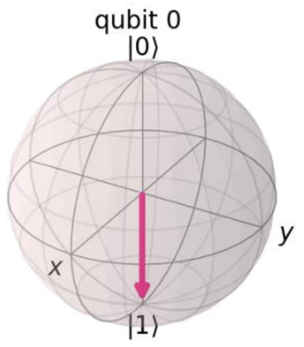
```
initial_state.equiv(final_state)
```

```
False
```

```
initial_state.draw(output = 'bloch')
```



```
final_state.draw(output = 'bloch')
```



Analysis and Results

From the Implementation section, it is evident that phase-flip code cannot correct bit-flip errors and the phase-flip code cannot correct bit-flip errors. But Shor code can correct both types of errors and preserve the initial state by detecting and correcting any errors that occur. However, Shor code has its limitations. It fails to correct the errors when the probability of errors occurring in the main qubit or other qubits is very high.

State fidelity:

It is an important measurement in quantum physics in analyzing error correction simulations. It measures how close two quantum states are to each other on a scale of 0 (completely different) to 1 (the same). Increasing the probability of errors occurring leads to a higher chance of the final state being different from the initial state. Therefore, the state fidelity is inversely proportional to probability of errors.

- State fidelity between two qubits both in the $|0\rangle$ state:

  ```
  state_fidelity([1, 0], [1, 0])
  ```

  ```
  1.0
  ```

- State fidelity between the $|0\rangle$ and $|1\rangle$ states:

  ```
  state_fidelity([1, 0], [0, 1])
  ```

  ```
  0.0
  ```

- State fidelity between the $|0\rangle$ and $|+\rangle$ states:

  ```
  import math
  state_fidelity([1,0], [1/math.sqrt(2),1/math.sqrt(2)])
  ```

  ```
  0.4999999999999999
  ```

- State fidelity between the initial and final states for a Shor code where the probability of errors occurring is 30%.

  ```
  state_fidelity(initial_state, final_state)
  ```

  ```
  1.0
  ```

- State fidelity between the initial and final states for a Shor code where the probability of errors occurring is 70%.

  ```
  state_fidelity(initial_state, final_state)
  ```

  ```
  3.680050034621245e-103
  ```

  The state fidelity is an extremely small number which is equivalent to zero.

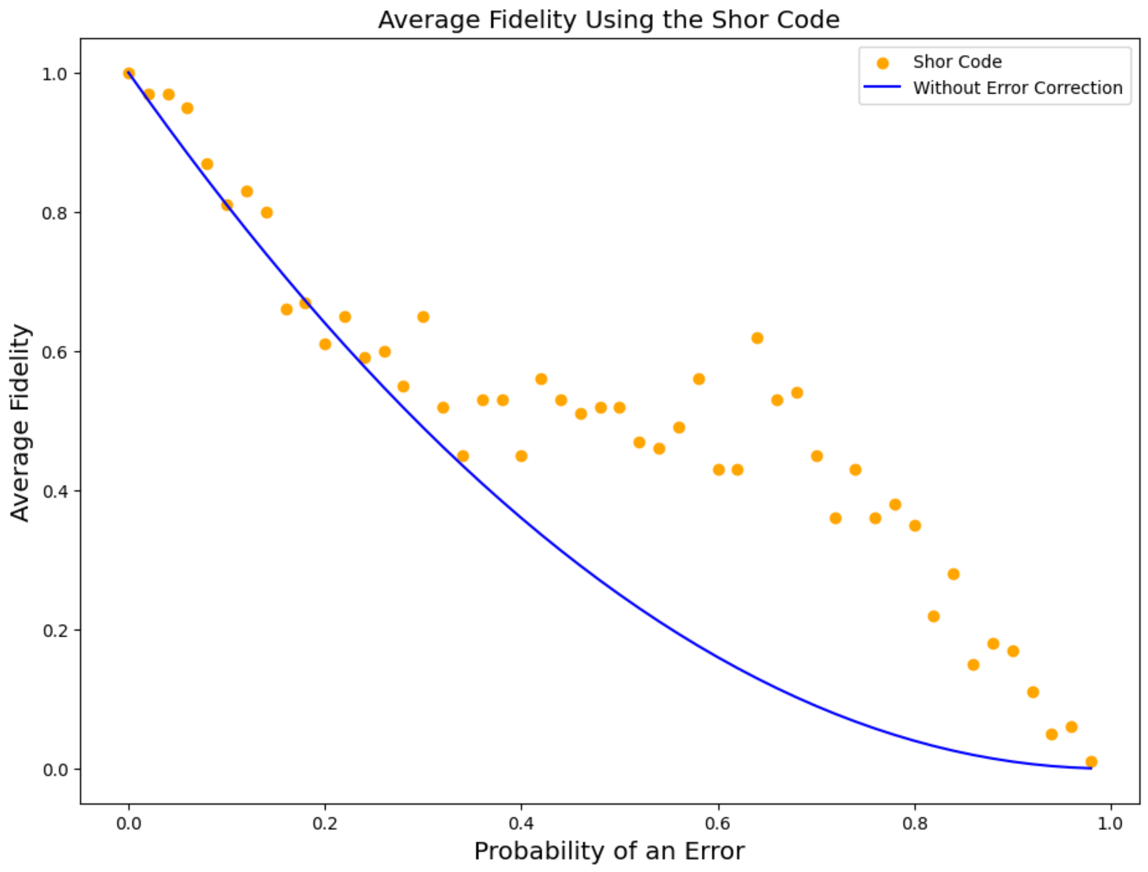
Limitations of Shor Code:

- Applying excessive Z gates combined with Hadamard gates to simulate errors can cause the qubits in the system to change phases, enter superpositions and get entangled with other qubits or systems which causes instability in the system and Shor code fails to correct errors. The error is defined in Qiskit as QiskitError: 'Density matrix is not a pure state'. It is a sign of change in the quantum state so serious that the qiskit function for creating the statevector effectively fails. In quantum physics, this particular change is called going from a "pure state" to a "mixed state".

- Since each logical qubit requires nine physical qubits, to encode n logical qubits we require 9n physical qubits. Once we have encoded multiple logical qubits, we would need to perform error correction on each of them separately. Implementing the Shor code with multiple qubits would require a more complex error correction procedure and numerous physical qubits. As of now in May 2023, the largest quantum computer has 433 qubits and therefore encoding large number of logical qubits currently isn't feasible. However, in future, when quantum computers advance, implementing QEC codes on a large scale will be possible.

Comparison of Shor code to no error correction:

The following graph is of average state fidelities with increasing probability of errors.

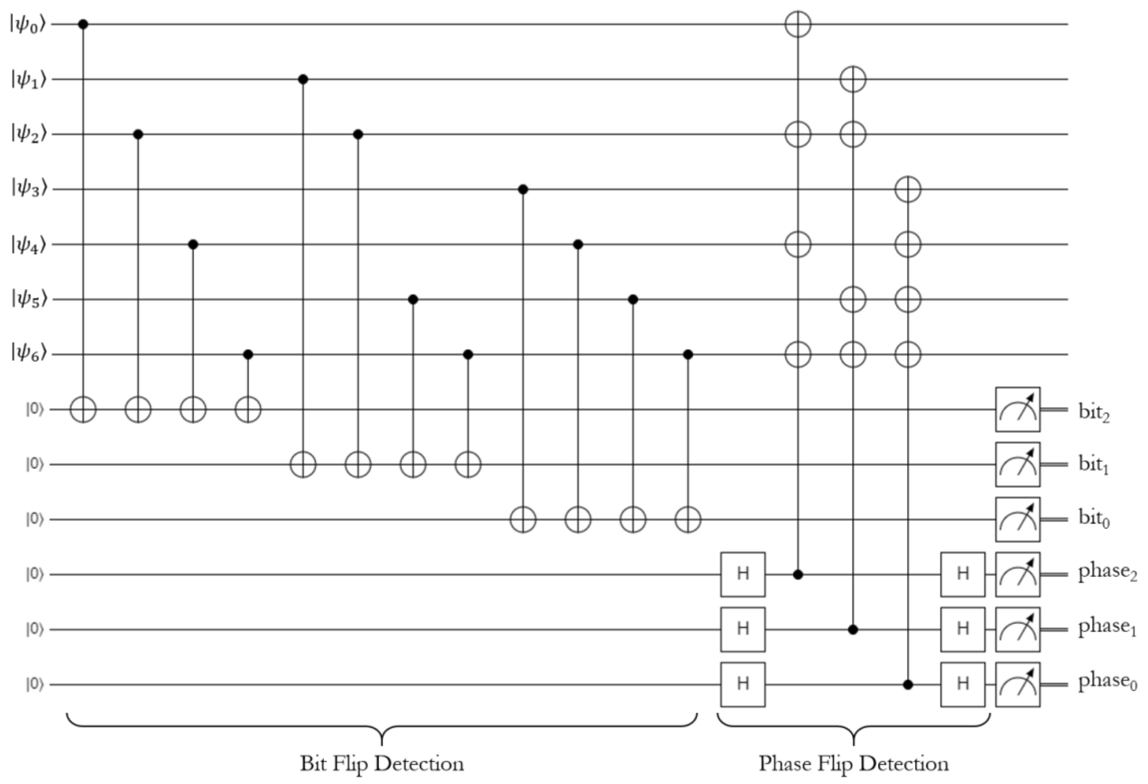


Observations based on the above graph:

- Initially, for probabilities of errors occurring between 0% and 30%, the state fidelity of Shor code better than or almost similar to that of using no error correction. For this range Shor code's fidelity is 1 or closer to 1 which indicates the initial and final states were preserved after correcting errors.
- As the probability reaches around 30% to 70%, the Shor code's average fidelity almost forms a plateau in its curve, whereas the average fidelity for a system without error correction continues its downward trend. This indicates that Shor code's performance is better, and it still has a high chance to mitigate errors. The unusual plateau in the curve is because the introduction of errors based on probability, is still random and in case lower number of errors occur, they can still be corrected by Shor code and the fidelity doesn't drop down immediately.
- After probability of around 70%, the curve of Shor code's state fidelity still stays above the curve of a system without error correction however, it follows a downward trend.

An alternative to Shor code:

Steane's Error Correction Code:
This QEC code is similar to Shor code, and it protects a qubit against phase-flip or bit-flip errors with a logical qubit created out of 7 physical qubits, where 6 qubits are spares. This code can correct both errors even if they occur on different qubits. The circuit is as follows:



References

- https://arxiv.org/pdf/0905.2794.pdf
- https://journals.aps.org/pra/abstract/10.1103/PhysRevA.52.R2493
- https://quantumcomputinguk.org/tutorials/quantum-error-correction-shor-code-in-qiskit
- https://stem.mitre.org/quantum/error-correction-codes/steane-ecc.html
- https://en.wikipedia.org/wiki/Quantum_computing
- https://quantum-computing.ibm.com/lab/docs/iql/tutorials-overview
- https://commons.wikimedia.org/w/index.php?curid=5829358
- https://qiskit.org/
- https://research.ibm.com/blog/ibm-quantum-roadmap-2025
- https://cs.uwaterloo.ca/~watrous/QC-notes/QC-notes.16.pdf
- https://en.wikipedia.org/wiki/Quantum_error_correction
- https://iopscience.iop.org/article/10.1088/0034-4885/76/7/076001
- https://journals.aps.org/prx/abstract/10.1103/PhysRevX.11.041058