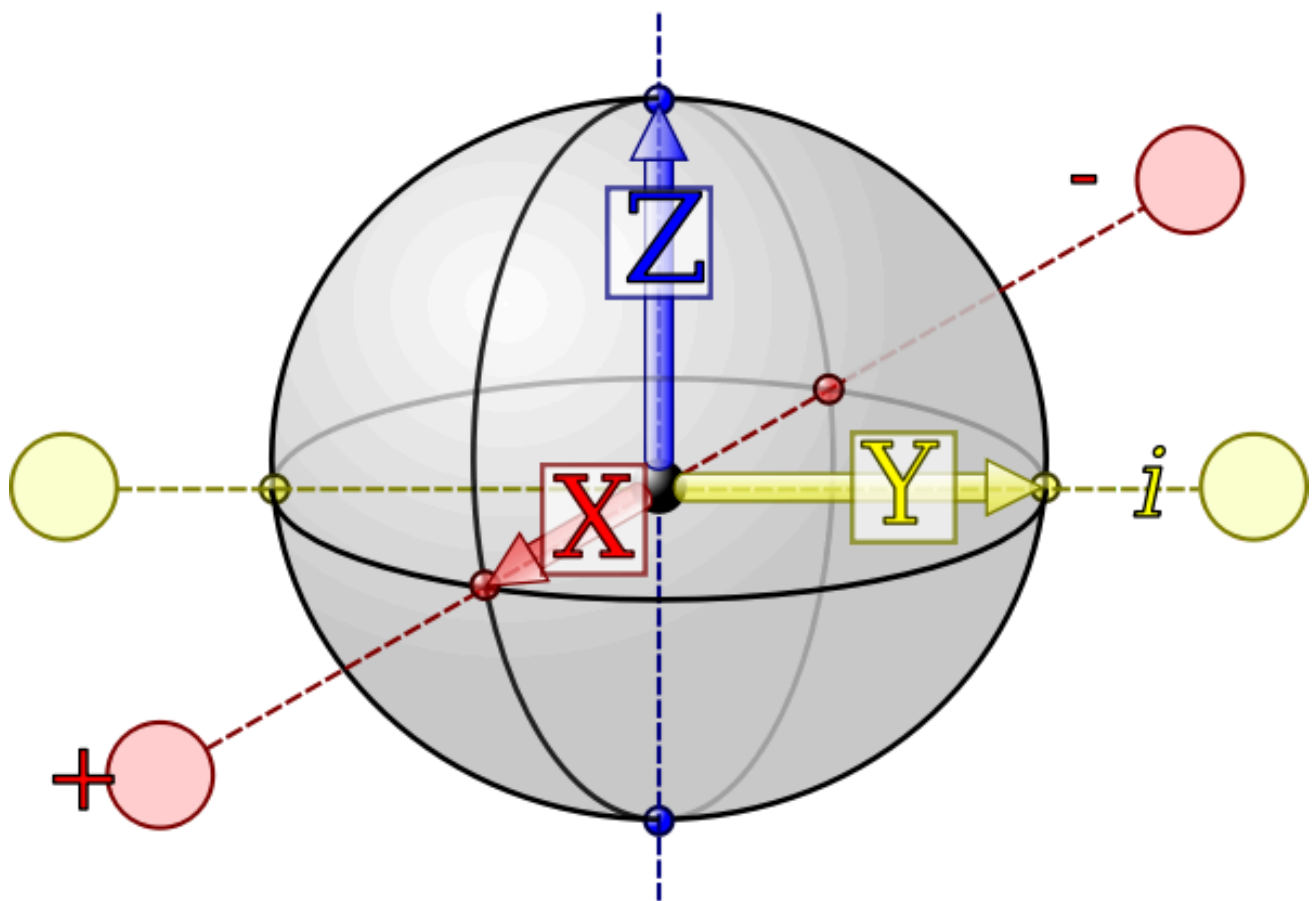


Implementation of Quantum Error Detection Circuits for Bit-Flip and Phase-Flip Errors on Two Qubits Using Qiskit



Introduction

Quantum mechanics, the foundation of quantum computing, fundamentally redefines computation by leveraging the unique principles of quantum states. Unlike classical bits, which can exist solely as 0 or 1, quantum bits or **qubits** possess properties that enable them to be both 0 and 1 simultaneously—a phenomenon known as **superposition**.

In quantum computing, errors such as **bit-flip** and **phase-flip** can occur due to environmental factors or imperfections in hardware. We present the design and implementation of error detection circuits for two types of errors (bit-flip and phase-flip) on two qubits using Qiskit, a popular quantum computing framework.

Crucial Properties of Quantum Information

➤ Superposition

- A qubit can exist in a superposition of states, represented by a linear combination of the basis states $|0\rangle$ and $|1\rangle$.

- Mathematically, a qubit's state is expressed as:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$$

where α and β are complex numbers such that $(|\alpha|^2 + |\beta|^2 = 1)$.

- This allows qubits to hold and process more information than classical bits.

➤ Entanglement

- When two or more qubits are entangled, their states become interdependent, forming correlations that hold even across large distances.

- Entangled qubits are critical in quantum algorithms for parallelism and faster computation.

➤ **Quantum Interference**







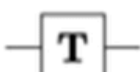

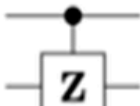
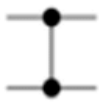


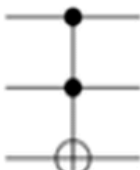
- Quantum interference arises from the probabilistic nature of qubit states and is used to amplify the probability of correct outcomes while diminishing erroneous paths.
- Manipulating interference is a key feature in optimizing quantum algorithms.

Key Terminologies in Quantum Mechanics

- **Qubit:** The fundamental unit of quantum information, analogous to a classical bit but capable of existing in superpositions of 0 and 1 states.
- **Measurement:** The process of observing a qubit's state, collapsing it from superposition to either $|0\rangle$ or $|1\rangle$ probabilistically.
- **Bloch Sphere:** A geometric representation of a qubit's state as a point on a sphere, useful for visualizing transformations.
- **Quantum Circuit:** A sequence of quantum gates applied to qubits, representing a computation in quantum computing.

Understanding these fundamental properties, matrices, and terminologies lays the groundwork for exploring advanced quantum algorithms, error correction methods, and practical applications in quantum computing.

Important Matrices in Quantum Computation

Pauli-X (X)			$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)			$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)			$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)			$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)			$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)			$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)			$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)	 		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP	 		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)			$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

Quantum Error Detection Circuits

Building circuits to detect bit-flip and phase-flip errors in quantum systems is essential for maintaining qubit fidelity, especially in quantum error correction.

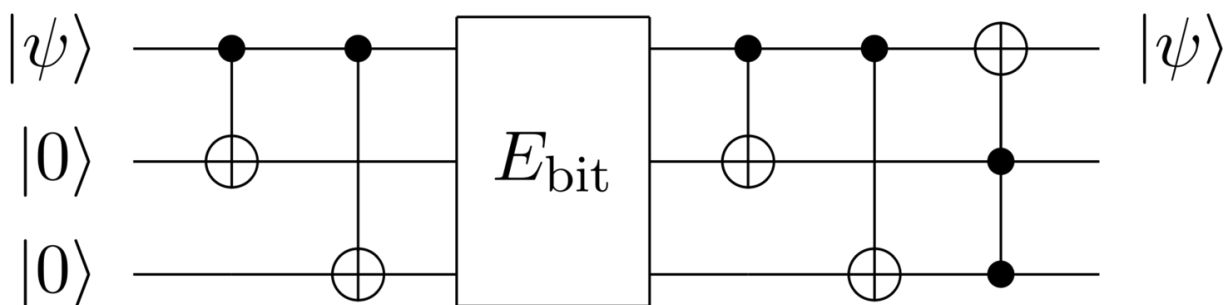
In this project, we focus on designing circuits that can identify these errors on two qubits, using foundational concepts like ***superposition, entanglement, and quantum gates***. Specifically, we leverage ***Pauli matrices*** (X, Y, and Z gates) to introduce and detect errors, as well as the ***Hadamard gate*** for creating and measuring phase changes.

By employing ***syndrome measurement*** techniques, we can observe specific states that indicate errors, offering a pathway to safeguard quantum computations against noise and interference.

BIT FLIP

What is a Bit Flip Error?

A bit flip error changes the qubit's state from 0 to 1 or vice versa, like an ***X-gate***. It is corrected using the bit flip code, which copies the qubit's state to two ancillary qubits via ***CNOT gates***. If an error occurs, it is detected by measuring discrepancies between the qubits. The main qubit is then corrected using *additional CNOT and Toffoli gates* based on the ancillary qubits' states.

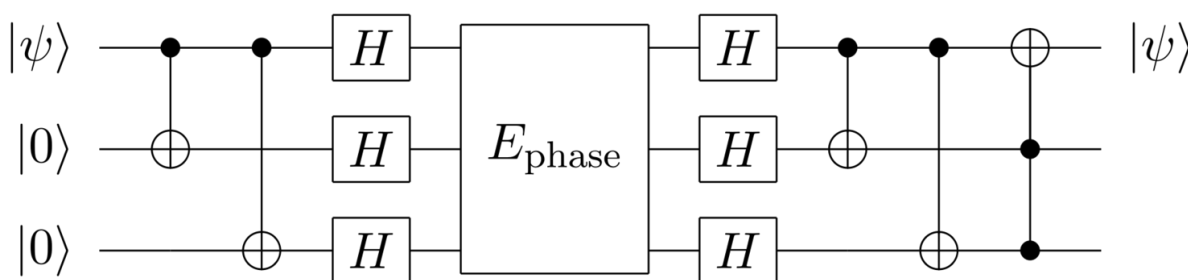


Circuit diagram of the Bit flip code

PHASE FLIP

What is a Phase Flip Error?

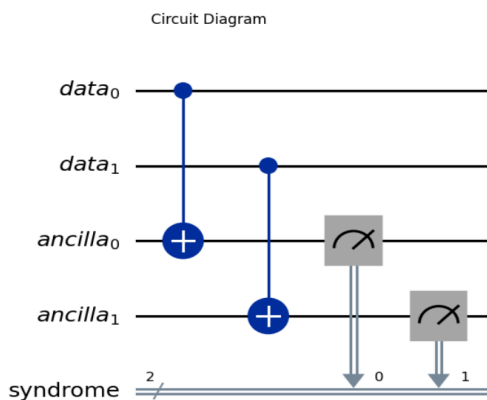
A phase flip error alters the qubit's phase, similar to a **Z-gate**. It can be corrected using the phase flip code, which involves transferring the main qubit's state to ancillary qubits with **CNOT gates**, followed by **Hadamard** gates to *create superposition*. After the error, Hadamard gates are applied again to revert superposition. Finally, the error is corrected using CNOT and Toffoli gates with the ancillary qubits as controls.



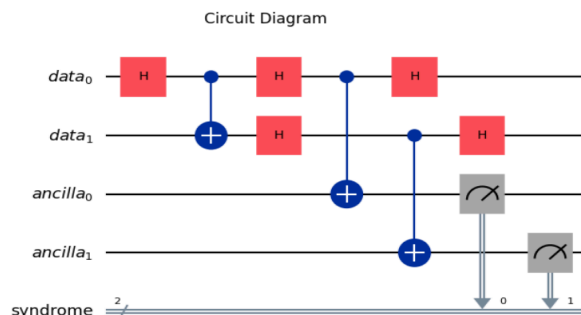
Circuit diagram of the phase flip code

ABOUT ERROR DETECTION CODE

The code for error detection implements a quantum error detection system using the Qiskit library. Specifically, it creates and simulates quantum circuits to detect bit-flip and phase-flip errors in quantum bits (qubits).



Bit-Flip Error Detection



Phase-Flip Error Detection

FUNCTION OVERVIEW:

`flip_detection(qc):`

- Adds gates to a circuit to detect bit-flip errors. It uses two ancillary (syndrome) qubits to check for errors on two data qubits.

`phase_detection(qc):`

- Detects phase-flip errors by converting the data qubits to the X-basis, performing error detection similar to the bit-flip method, and converting them back to the Z-basis.

`create_bit_flip_detection_circuit()` and `create_phase_flip_detection_circuit():`

- Each creates a circuit layout for detecting bit-flip or phase-flip errors, setting up two data qubits and two ancilla qubits, with a classical register to store measurement results.

`simulate_error_detection(circuit, error_type='bit_flip'):`

- Simulates the error detection circuit on a quantum simulator (QASM) and runs measurements. The error type parameter ('bit_flip' or 'phase_flip') determines which detection function (bit-flip or phase-flip) is used.

visualize_circuit_and_results(circuit, counts, title):

- Visualizes the quantum circuit and measurement results using Matplotlib. The circuit diagram and a histogram of measurement outcomes are displayed side by side.

demonstrate_error_detection():

- Demonstrates bit-flip and phase-flip error detection. It first shows results without errors and then applies a bit-flip or phase-flip error on specific qubits to show detection outcomes.

analyze_circuit_details(circuit):

- Prints details of a quantum circuit, such as the number of qubits, classical bits, circuit depth, and a breakdown of operations. Useful for understanding circuit complexity.

DETECTION + CORRECTION CODE

Qiskit Code Implementation for Error Correction

```
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister, transpile
from qiskit_aer import Aer
import matplotlib.pyplot as plt
from qiskit.visualization import plot_circuit_layout
```

- Importing libraries and dependencies.

```
def add_cnot(qc, q1, q2, q3):
    qc.cx(q1, q2)
    qc.cx(q1, q3)

    return qc
```

- Function to add CNOT gates from qubit 1 to 2 and 1 to 3 respectively. Returns modified quantum circuit.

```
def add_toffoli(qc, q1, q2, q3):
    qc.ccx(q3, q2, q1)
    return qc
```

- Function to add a Toffoli Gate to q1 with q2 and q3 as the control gate. Returns modified quantum circuit.

```
def add_hadamard(qc, q1, q2, q3):
    qc.h(q1)
    qc.h(q2)
    qc.h(q3)

    return qc
```

- Applies a Hadamard gate to each qubit. Returns modified quantum circuit.

Bit Flip Function

```
def bit_flip_correction():  
    q = QuantumRegister(2, 'q')  
    a = QuantumRegister(4, 'a')  
    c = ClassicalRegister(2, 'c')  
  
    qc = QuantumCircuit(q, a, c)
```

- Defining and initializing a quantum circuit for bit flip correction with 2 main qubits, 4 auxiliary qubits and a classical register with 2 classical bits.

```
###  
  
# Create any 2 qubit state here using q[0] and q[1]  
# qc.h(q[0])  
# qc.cx(q[0], q[1])  
  
###
```

- Code to use entangled state qubits instead of default $|0\rangle$ state..

```
qc = add_cnot(qc, q[0], a[0], a[1])  
qc = add_cnot(qc, q[1], a[2], a[3])  
  
#### Introducing bit flip errors here  
  
# qc.x(q[0])  
# qc.x(q[1])  
  
####  
  
qc = add_cnot(qc, q[0], a[0], a[1])  
qc = add_cnot(qc, q[1], a[2], a[3])  
  
qc = add_toffoli(qc, q[0], a[0], a[1])  
qc = add_toffoli(qc, q[1], a[2], a[3])
```

- Adding CNOT and Toffoli gates in the quantum circuit.

-
- The first set of CNOT gates entangles the states of q0 and 1 with a0, a1 and a2, a3, respectively.
 - The commented out code can be uncommented to add bit flip errors.
 - The second set of CNOT is applied to compare our main qubits with their auxiliary qubits to check if any errors have occurred. [Detection]
 - The Toffoli gates flip the main qubits on the basis of the auxiliary qubits to correct them. [Correction]

```
qc.measure([q[0],q[1]], [c[0],c[1]])

simulator = Aer.get_backend('qasm_simulator')
result = simulator.run(qc, shots=1024).result()
counts = result.get_counts()
print("After Bit Flip Error Detection and Correction Counts:", counts)
```

- We then measure the qubit states and store their values in the classical states.
- We use the qasm_simulator to run our circuit for 1024 shots [repeated runs of the circuit]. We then output the number of result counts we had for error detection and correction.

```
fig, axs = plt.subplots(1, 1, figsize=(12, 4))
qc.draw(output='mpl', ax=axs)

plt.tight_layout()
plt.show()
return
```

- Using matplotlib to plot our circuit.

Phase Flip Function

```
def phase_flip_correction():
    q = QuantumRegister(2, 'q')
    a = QuantumRegister(4, 'a')
    c = ClassicalRegister(2, 'c')

    qc = QuantumCircuit(q, a, c)
```

- Similar declaration of qubits and classical bits as bit flip.

```
###

# Create any 2 qubit state here using q[0] and q[1]
# qc.h(q[0])
# qc.cx(q[0], q[1])

###
```

- In case, we want to use entangled bell states as our qubits.

```
qc=add_cnot(qc,q[0],a[0],a[1])
qc= add_cnot(qc,q[1],a[2],a[3])

qc = add_hadamard(qc,q[0],a[0],a[1])
qc= add_hadamard(qc,q[1],a[2],a[3])
```

- Adding CNOT and Hadamard gates to the circuit.
- The key action we are doing here by applying the Hadamard gates is changing the phase flip errors into bit flip errors.

```
#### Introducing phase flip errors here

# qc.z(q[0])
# qc.x(q[1])

####
```

- Uncomment to introduce phase flip errors.

```

qc = add_hadamard(qc,q[0],a[0],a[1])
qc= add_hadamard(qc,q[1],a[2],a[3])

qc=add_cnot(qc,q[0],a[0],a[1])
qc =add_cnot(qc,q[1],a[2],a[3])

qc = add_toffoli(qc,q[0],a[0],a[1])
qc = add_toffoli(qc,q[1],a[2],a[3])

```

- Adding Hadamard, CNOT and Toffoli gates to our quantum circuit.
- The second round of Hadamard gates reverts the error back to the original basis.
- The CNOT and Toffoli gates work to detect the error and correct it. This works similar to how it was in bit-flip error detection and correction.

```

qc.measure([q[0],q[1]], [c[0],c[1]])

simulator = Aer.get_backend('qasm_simulator')
result = simulator.run(qc, shots=1024).result()
counts = result.get_counts()
print("After Phase Flip Error Detection and Correction Counts:", counts)

fig, axs = plt.subplots(1, 1, figsize=(12, 4))
qc.draw(output='mpl', ax=axs)

plt.tight_layout()
plt.show()

```

- We then measure the quantum states and save them in the classical bits.
- The simulation is run again and the circuit diagram is plotted.

```

if __name__ == "__main__":
    bit_flip_correction()
    phase_flip_correction()

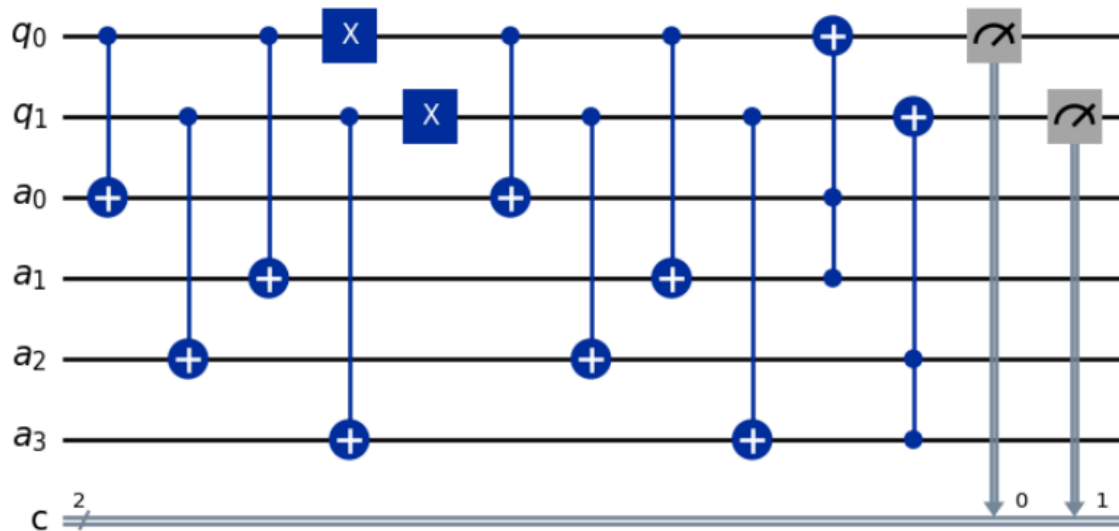
```

- A condition checks if the script is being run. If it is, it then calls both the functions.

OUTPUTS

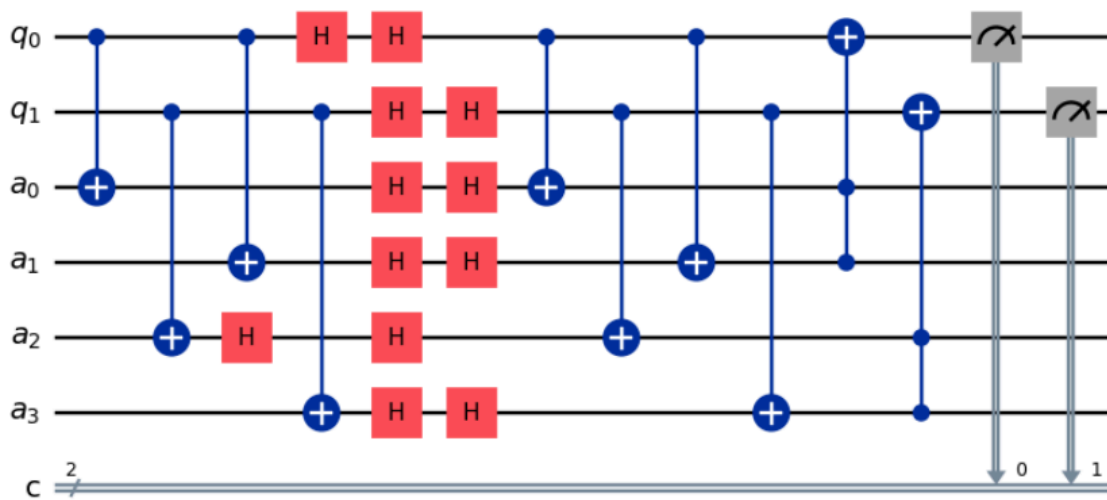
Bit Flip Detection and Correction Output

After Bit Flip Error Detection and Correction Counts: {'00': 1024}



Phase Flip Detection and Correction Output

After Phase Flip Error Detection and Correction Counts: {'00': 1024}



Conclusion

In this project, we have developed and analyzed quantum circuits for **detecting and correcting** bit-flip and phase-flip errors on two qubits. We have also incorporated **error correction** alongside error detection to enhance the reliability and resilience of quantum computations.

The bit-flip and phase-flip errors are some of the most common forms of decoherence in quantum systems and can severely impact quantum computations if left unchecked. By leveraging syndrome measurements and error correction protocols, our circuits can identify and address these errors, preserving qubit fidelity.

Through practical implementation in *Qiskit*, we demonstrated that error detection is achievable using basic quantum gates and ancilla qubits. This work underscores the importance of error detection as a foundation for more advanced quantum error correction methods, crucial for the development of reliable quantum computing.

Real Life Future Applications :

Quantum error detection and correction are vital components of emerging quantum technologies. Some key applications include:

1. **Quantum Computing:** Error detection is essential for stable qubit operations, enabling accurate quantum computations in fields like cryptography and optimization.
2. **Quantum Communication:** In Quantum Key Distribution (QKD), error correction safeguards cryptographic keys, ensuring secure data transmission over long distances.
3. **Quantum Sensing:** Error detection preserves the accuracy of quantum sensors, crucial for applications in fields like medical imaging and navigation.
4. **Quantum Cryptography:** Quantum error correction secures cryptographic systems by protecting against data tampering and leakage, enabling next-gen secure communication.

Resources

- ❖ <https://quantumcomputinguk.org/tutorials/quantum-error-correction-phase-flip-code-in-qiskit>
- ❖ https://astro.pas.rochester.edu/~aquillen/phy265/lectures/QI_E.pdf
- ❖ <https://quantumcomputing.stackexchange.com/questions/21492/can-error-correction-for-a-classical-algorithm-with-bit-flips-be-easier-than-for>
- ❖ <https://quantumcomputinguk.org/tutorials/quantum-error-correction-bit-flip-code-in-qiskit>