Convergence
A Journal for Young Researchers

ORIGINAL REVIEW

Received: 11-19-25; Accepted: 3-2-26;
Managing Editor: Jonas Katona;
Action Editor: George Zifkos

An Overview Of Assuring Fault Tolerance In Quantum Computers Through Quantum Error Correction

Raghav Sriram

Victoria Junior College, Singapore

Abstract

Quantum computers promise to solve computational problems that are intractable for classical systems, but their inherent sensitivity to noise and decoherence can render the results of any quantum computation useless. Without mitigation, small errors can accumulate, leading to the corruption of quantum information and failure of computations. Fault-tolerant quantum computation is a framework that provides us with tools to reduce the error in any operation performed by a Quantum Computer. This paper presents an overview of fault-tolerant quantum computation, beginning with the foundational principles of quantum information and quantum error correction, and is written for undergraduates and non-specialists seeking a unified introduction to the field. We explore the mathematical structure of noise models, introduce quantum error correction codes such as the three-qubit bit-flip code, and discuss techniques like syndrome measurement and error detection. We then develop the framework of fault tolerance, highlighting the role of transversal gates and the threshold theorem, which ensures that reliable computation is possible if the physical error rate is below a certain threshold. Finally, we outline current challenges in reducing overhead and improving error correction efficiency, pointing to future directions in the pursuit of scalable and reliable quantum technologies. This paper provides a comprehensive foundation in fault-tolerant quantum computation while offering insights into the future directions of this field.

Keywords: fault-tolerant quantum computation, quantum error correction, stabilizer codes, surface codes, quantum noise models, quantum decoherence, threshold theorem, logical qubits, quantum fault tolerance

1. Introduction

Large-scale quantum computing is fundamentally limited by the fragility of quantum systems, in which errors arise from interactions with the environment, control imperfections, and decoherence. (Preskill, 1998; Nielsen & Chuang, 2010) These errors, which manifest as bit-flip errors, phase-flip errors, or a combination of both, can corrupt quantum information and

compromise the accuracy of results. (Shor, 1995; Devitt et al., 2013) Without ways to manage these errors, quantum algorithms fail to produce reliable results, making it impossible to scale quantum systems beyond a few qubits. (Preskill, 2018)

To address these challenges, quantum error correction (QEC) provides a framework for protecting quantum information by encoding a logical qubit into multiple physical qubits. (Gottesman, 1997; Devitt et al., 2013) QEC techniques, such as the three-qubit bit-flip code or the surface code, identify and correct errors through syndrome measurements, which detect the presence and type of errors without collapsing the quantum state. (Dennis et al., 2002; Fowler et al., 2012) However, error correction alone is insufficient. Without additional safeguards, error correction circuits themselves can introduce new errors, potentially rendering the error correction process useless. (Knill, 2005)

Fault-tolerant quantum computation (which will hereafter be referred to as FTQC) extends QEC by ensuring that errors do not spread uncontrollably through a quantum system. (Gottesman, 1998; Preskill, 1998) Fault tolerance specifically refers to performing computations reliably despite noise, given that error rates are below a threshold value. Fault-tolerant protocols, such as transversal gates that prevent error propagation and magic state distillation that enables the implementation of non-Clifford gates, are critical for preserving the integrity of quantum operations. (Bravyi & Kitaev, 2005; Reichardt, 2005) The threshold theorem guarantees that if the physical error rate is kept below a critical value, fault-tolerant protocols can reduce errors to very low levels, enabling reliable quantum computation over long durations. (Aharonov & Ben-Or, 2008)

As quantum hardware progresses beyond the limitations of noisy intermediate-scale quantum (NISQ) devices, achieving FTQC requires the development of architectures that integrate error correction with fault-tolerant protocols. (Preskill, 2018)

This paper is an introductory review on FTQC, aimed at providing early-stage researchers and undergraduates with an accessible bridge to the field's formalisms. This review adds a unified presentation of QEC, where one consistent and clear set of assumptions and examples is carried from noise channels through stabilizer syndrome projectors, error propagation identities, transversal constructions, and the threshold theorem. In particular, the channel-based noise models in Section 3 are carried forward as Pauli error processes that define stabilizer syndromes and enable the later analysis of error propagation, transversal gates, and the threshold theorem. Furthermore, this review contributes an educational Qiskit simulation that guides readers step by step through syndrome extraction, diagnosis, and recovery for a noisy three-qubit code.

2. Literature Review

QEC emerged as a response to the inherent fragility of quantum information. Early schemes like the Shor code (Shor, 1995) and the stabilizer formalism developed by Gottesman (1997) laid the foundation for encoding logical qubits into entangled states of multiple physical qubits. These encodings allow errors to be detected and corrected through syndrome measurements without disturbing the encoded information.

Topological codes, especially the surface code introduced by Dennis et al. (2002), became a leading approach due to their high error thresholds and compatibility with 2D qubit architectures. Fowler et al. (2012) expanded this work with practical implementations, establishing the surface code as a viable candidate for scalable quantum computing.

To support reliable quantum operations, fault-tolerant protocols are essential. Aharonov and Ben-Or (2008) and Gottesman (1998) introduced the threshold theorem, showing that errors can be suppressed below any desired level if the physical error rate is beneath a critical threshold. However, achieving universal computation requires non-Clifford gates, which are not fault-tolerantly implementable through transversal means. Bravyi and Kitaev (2005) addressed this limitation through magic state distillation, enabling universal gate sets within a fault-tolerant framework. Magic states are needed for universality, as no quantum error-correcting code has a universal set of transversal gates (Eastin–Knill Limitation).

Efforts to reduce the overhead of fault tolerance have explored alternative error-correcting codes. Subsystem codes (Poulin, 2005), bosonic codes (Michael et al., 2016), and hardware-specific models (Geller & Zhou, 2020) offer more efficient schemes under realistic noise. Despite these advances, challenges remain in lowering qubit requirements and developing fault-tolerant architectures that can operate under experimental noise conditions.

3. QUANTUM NOISE AND DECOHERENCE

3.1. Introduction to Quantum Noise

Quantum systems, as we have already mentioned, are inherently fragile. Unlike their classical counterparts, where copying and redundancy can be used to mitigate errors, quantum systems are subject to noise arising from interactions with their environment. These disturbances lead to decoherence, which is the gradual destruction of quantum superposition and entanglement. Understanding the origins and modelling of quantum noise is essential to the development of FTQC, where errors must be corrected without disturbing the logical state of the system.

Noise in quantum systems arises because no quantum system is perfectly isolated. Superconducting circuits, trapped ions, and spin systems are all inevitably coupled to their surrounding environment. These interactions can cause uncontrolled evolution of a quantum state, which causes phenomena such as amplitude damping, phase damping, or random bit flips. Unlike classical errors, quantum errors may involve arbitrary rotations in Hilbert space. Moreover, due to the no-cloning theorem, quantum information cannot be copied and restored from backups. Therefore, the noise within the system must be addressed by embedding the quantum information into larger systems and detecting errors indirectly, without observing the logical qubit itself. To model this, we need to turn to the formalism of quantum channels, which provides us with a mathematical description of noisy evolutions using the operator-sum representation.

3.2. Quantum Basics

Before introducing quantum noise through Kraus operators, it is useful to briefly establish several foundational concepts that clarify how quantum states are represented, how errors act on them, and why stabilizer measurements do not destroy the encoded information. This section serves as a conceptual bridge between the formal description of qubits and the mathematics behind QEC, particularly for those with less experience in quantum computation. A qubit state is most simply described as a pure state, written in Dirac notation as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \text{ where } \alpha, \beta \in \mathbb{C} \text{ and } |\alpha|^2 + |\beta|^2 = 1$$

Pure states represent maximal knowledge about a quantum system. However, in practice, a qubit is often not prepared in

the same state each time, or it becomes entangled with an unobserved environment. In these situations, the qubit cannot be described by a single state vector |⟩ alone, because our description must also include classical uncertainty and environmental interference. The appropriate representation is the density matrix. For a pure state, the density matrix is

$$\rho = |\psi\rangle\langle\psi|$$

whereas a mixed state is a probabilistic ensemble {(pi|i)} described by

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$$

Mixed states, therefore, capture situations where the system is not in any single definite state from our perspective. They are the standard language for modeling noise processes such as decoherence and dissipation.

Errors acting on qubits are commonly expressed using the Pauli operators:

The Pauli-X Gate, where $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$, which is effectively a bit flip.

The Pauli-Z Gate, where $Z|0\rangle = |0\rangle$ and $Z|1\rangle = -|1\rangle$, which corresponds to a phase flip.

The Pauli-Y Gate, where $Y = iXZ$, which performs both a bit and phase flip.

Importantly, any arbitrary single-qubit error can be expressed as a linear combination of these Pauli operators, which is why Pauli errors form the basis of most quantum error-correction frameworks.

One of the largest concerns in quantum computing is extracting information about errors without learning the quantum state itself, which would cause the state's superposition to collapse. This can be achieved through stabilizer measurements. Stabilizer operations are chosen such that all valid code states are eigenstates with eigenvalue +1. Measuring a stabilizer, therefore, reveals if an error has occurred without distinguishing between different logical states in the code space. As a result, error syndromes can be extracted repeatedly without collapsing the logical qubit.

With this groundwork set, we are prepared to describe quantum noise more formally using the density-matrix framework and Kraus operator representations in the following section.

3.3. Quantum Channels and Kraus Operators

A quantum channel is a completely positive, trace-preserving (CPTP) linear map. A CPTP map is a mathematical operation that describes the most general, physically allowed transformation of a quantum state, ensuring that it remains a valid density matrix even when extended to larger entangled systems. In this case, a Quantum Channel can be described as:

$$\varepsilon: \mathfrak{B}(\boldsymbol{\mathcal{H}}) \to \mathfrak{B}(\boldsymbol{\mathcal{H}}),$$

where $\mathfrak{B}(\boldsymbol{\mathcal{H}})$ denotes the space of bounded operators on the Hilbert space $\boldsymbol{\mathcal{H}}$. Physically, this map takes a density matrix representing the state of a quantum system and maps it to another valid density matrix, accounting for the possibilities of

external interaction from the environment.

For any quantum channel, three mathematical conditions must be fulfilled. Firstly, we have the condition of linearity:

$$\varepsilon(a\rho + b\sigma) = a\varepsilon(\rho) + b\varepsilon(\sigma), \text{ for any scalars } a, b \in \mathbb{C}$$

This means that if the input to a quantum channel is a linear combination of density matrices, then the output is also the linear combination of their individual outputs. Secondly, we have the condition of Trace Preservation:

$$Tr(\varepsilon(\rho)) = Tr(\rho)$$

This means that the trace of the density matrix is always one, which indicates that the total probability is 1 if the state is a pure state. If an operation changed the trace, it would mean that probability is either lost or created, which causes the value to be less than or more than 1, which is non-physical. Lastly, we have the condition of complete positivity:

$$\forall \rho_{AR} \geq 0, \ (\varepsilon \otimes I_n)(\rho_{AR}) \geq 0$$

$$\forall \rho_{AR} \geq 0, \ (\varepsilon \otimes I_n)(\rho_{AR}) \geq 0$$

This describes two entangled systems, A and R (of dimension n). The map is applied only to A and not R. Since this system is entangled, the resulting joint state must still be a valid quantum state. The resulting joint state must remain positive semidefinite. If a map is not completely positive, applying it to part of an entangled state could potentially produce negative eigenvalues, which can result in negative probabilities. For instance, consider the matrix transpose map, defined as ()=T. When applied to a non-entangled density matrix, the transpose operation preserves positivity and appears harmless. However, it is not completely positive. If we apply I to one half of a maximally entangled Bell state, the resulting joint state is no longer positive semidefinite, and it acquires negative eigenvalues.

A very popular representation of a quantum channel is its operator-sum decomposition, otherwise known as the Kraus representation. Any CPTP map can be written as:

$$\varepsilon(\rho) = \sum_k E_k \rho E_k^\dagger$$

where the operators Ek are Kraus Operators, which satisfy the completeness condition:

$$I = \sum_k E_k^\dagger E_k$$

This ensures that the total transformation preserves the trace. Each Kraus operator can be interpreted as corresponding to a possible transformation the system undergoes due to the interaction with the environment. Since we do not observe the environment, the final state is a weighted mixture of these outcomes.

3.4. Bit Flip Channel

One of the simplest examples of a quantum noise model would be the bit flip channel. It describes a situation in which a

qubit has some probability p of undergoing a Pauli-X operation, and a 1-p probability of remaining unchanged. The channel is defined as:

$$\varepsilon_{bit\ flip}(\rho) = (1 - p)\rho + pX\rho X$$

Alternatively, in Kraus form, it can be represented as:

$$\varepsilon(\rho) = E_0\rho E_0^\dagger + E_1\rho E_1^\dagger \text{, where } E_0 = \sqrt{1-p}\cdot I \text{ and } E_1 = \sqrt{p}\cdot X \text{, fulfilling } E_0^\dagger E_0 + E_1^\dagger E_1 = I$$

The bit flip channel captures the quantum take on a classical bit error, but unlike in the classical case, such noise affects superpositions as well. In FTQC, this bit flip channel serves as a foundational example in the construction of quantum error correcting codes. The three-qubit repetition code, as discussed in the next chapter, is designed to detect and correct such errors using redundancy and syndrome measurements.

4. BASICS OF QUANTUM ERROR CORRECTION

4.1. Introduction to Stabilizer Codes

Quantum Error Correction allows us to detect and correct errors that result from environmental interference. This is done by encoding a single logical qubit into multiple physical qubits in a way that allows error detection and correction through carefully chosen measurements. Here, we will introduce the foundations of QEC through the framework of stabilizer codes.

Stabilizer operators are operators that do nothing to a quantum state. In the given example:

$$S|\psi\rangle = |\psi\rangle.$$

We can say that S stabilises the state |⟩, and that the state provides a +1 eigenvalue of the operator S. Stabilizer operators are commonly made from Pauli Matrices.

We can take the tensor products of these to get operators that can be applied to multiple qubits. For instance $Z_1Z_2 = Z \otimes Z \otimes I$, or $X_1X_3 = X \otimes I \otimes X$. (In this paper, we will be dealing with and simulating purely 3-qubit systems. As such, the aforementioned notation exclusively describes a 3-qubit system.)

In stabilizer quantum error correction, we define a code space by specifying a set of operators that leave the valid quantum states unchanged. These operators form what is called a stabilizer group, usually denoted S, and are chosen from the Pauli group over n qubits:

$$P_n = \{\pm 1, \pm i\} \cdot \{I, X, Y, Z\}^{\otimes n}.$$

The definition of the code space C can be formally denoted as such:

$$C = \left\{ |\psi\rangle \in \mathbb{C}^{2^n} \mid S|\psi\rangle = |\psi\rangle \ \ for\ all\ S \in \boldsymbol{S} \right\}.$$

This essentially means that the code space C is the set of all quantum states |⟩ in the n-qubit Hilbert space such that each Stabilizer leaves |⟩ unchanged.

Now, suppose some error operator EPn acts on a valid codeword |⟩C. The resulting state is E|⟩, which may or may not lie in the code space anymore. If we add a stabilizer to this new state:

$$S\left(E|\psi\rangle\right) = \begin{cases} +E|\psi\rangle & if\ [S,E] = 0\ (they\, commute) \\ -E|\psi\rangle & if\ \{S,E\} = 0\ (they\, anti-commute) \end{cases}$$

If E commutes with all stabilizers, then E|⟩ is still stabilised by S, meaning the error is undetectable by the stabilizer measurements. These types of errors are either trivial or logical operations: they map one valid codeword to another in the code space. If E anti-commutes with at least one stabilizer, then the state E|⟩ is no longer stabilised by that operator. Measuring that stabilizer will return -1 instead of +1, flagging the error. This flips the syndrome associated with the codeword. The stabilizer acts like a signature of what error occurred, allowing the system to infer what went wrong without measuring or collapsing the encoded quantum state.

A stabilizer code is often described by 3 numbers: n, the number of physical qubits, k, the number of logical qubits encoded, and d, the minimum number of qubit errors needed to confuse one code word with another. One example of this would be the Steane code, which can be represented as [[7,1,3]], which can be used to correct one arbitrary qubit error.

4.2. The Three-Qubit Bit-Flip Code

We will now examine the three-qubit bit-flip code as a simple yet illustrative example of a stabilizer code. The three-qubit bit-flip code protects a single logical qubit from a single bit-flip error by encoding it into three physical qubits using repetition. Despite its simplicity, the three-qubit code demonstrates the fundamental principles of quantum error correction.

Given a logical qubit:

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

the encoded state is:

$$|\psi\rangle_L = \alpha|000\rangle + \beta|111\rangle.$$

The state stores the logical value, and all 3 qubits are in agreement. A bit flip on any of the qubits can take the state out of the code space:

$$X_1|\psi\rangle_L = \alpha|100\rangle + \beta|011\rangle$$
$$X_2|\psi\rangle_L = \alpha|010\rangle + \beta|101\rangle$$
$$X_3|\psi\rangle_L = \alpha|001\rangle + \beta|110\rangle$$

In each case, the state becomes orthogonal to the original code space, but it remains distinguishable due to the specific error pattern. We can utilise this to detect and correct the error using syndrome measurements. As a stabilizer code, this code is defined by 2 generators:

$$S_1 = Z_1Z_2, \quad S_2 = Z_2Z_3$$

The code space is the +1 eigenspace of both stabilizers. Any bit-flip error will anticommute with one or both stabilizers, which can cause one or both syndrome values to flip to -1.

It is important to emphasize the scope of the three-qubit bit-flip code. The code is designed to correct at most one bit-flip (Pauli-X) error occurring on any of the three physical qubits. However, it does not protect against phase-flip (Pauli-Z) errors, since such errors commute with the stabilizer generators and therefore do not produce a detectable syndrome. Correcting both kinds of errors requires more powerful quantum error correction codes, such as the Shor code, the Steane code, or topological constructions like the surface code, which can address additional single-qubit errors and form the basis of fully fault-tolerant quantum computation.

4.3. Basic Syndrome Measurement and Error Correction

To identify and correct errors, we perform syndrome measurements, which involve measuring the stabilizer operators without causing a collapse of the encoded quantum information.

Firstly, we shall go over Syndrome Extraction. We measure S1=Z1Z2 and S2=Z2Z3. These two observables commute with each other and with the encoded state so their measurement outcomes reveal information about where a bit-flip could have occurred, but not about the encoded probability amplitudes. Using the previous example:

Table 1: Syndrome Extraction Table

Error	$S_1 = Z_1Z_2$	$S_2 = Z_2Z_3$	Syndrome
No Error	+1	+1	(0,0)
Flip on Qubit 1	-1	+1	(1,0)

Flip on Qubit 2	-1	-1	(1,1)
Flip on Qubit 3	+1	-1	(0,1)

Once the error is located via the syndrome, we apply the corresponding correction, which can be determined from the above table (Table 1). This procedure restores the state to the original codeword. The entire process can be repeated continuously to protect quantum information over time. Note that this code can only correct a single bit-flip error. This limitation reflects the code's distance of 3.

4.4. The Role of Ancillas in Syndrome Measurement

In quantum error correction, ancilla qubits play a vital role in non-destructively extracting error information from a quantum system. When measuring stabilizer operators, it is crucial to avoid collapsing or disturbing the encoded logical qubit. Directly measuring the data qubits would collapse their superposition and destroy the encoded quantum information. Ancilla qubits resolve this problem by acting as intermediaries. They are entangled with pairs of data qubits using a series of CNOT gates, in such a way that the parity of the data qubits is mapped onto the ancilla's state. The ancilla is then measured, revealing the syndrome without disturbing the logical state.

5. FAULT-TOLERANT QUANTUM COMPUTATION

5.1. Error Propagation and Catastrophic Failures

In classical circuits, errors can often be localised. However, quantum gates (particularly multi-qubit systems) can cause error propagation, which is when errors spread to multiple qubits in a single operation. This can often compromise the integrity of large amounts of quantum information, as even a single fault can corrupt many qubits.

Let us consider a CNOT gate acting on two qubits. If the control qubit has an X error, it remains on the control, and the error propagates to the target:

$$(X \otimes I)CNOT = CNOT(X \otimes X)$$

If the target qubit has a Z error, it can actually propagate back to the control qubit, causing errors to affect the entire system:

$$(I \otimes Z)CNOT = CNOT(Z \otimes Z)$$

This illustrates how error propagation may take place in a quantum system. Let us now consider a new situation where we have encoded a logical qubit using a stabilizer code that can correct one error per block (which is a group consisting of a logical qubit and the physical qubits that protect it). Suppose we apply a non-fault-tolerant CNOT gate between two of such blocks, and a single-qubit error occurs. If this error propagates into 2 errors within one code block, then the QEC code fails to correct it as it exceeds the error correction capacity of one error. This is what is known as Catastrophic

Failure. A more general description of Catastrophic Failures would be: a failure that occurs when multiple gates are applied across qubits in a block, and no mechanism is in place to limit the spread of errors, causing a single hardware fault to potentially invalidate the entire computation.

5.2. Transversal Gates and Error Suppression

A key tool in FTQC is the use of Transversal Gates. These gates are used to localise errors, preventing the spread of a single physical error into multiple qubits within the same block, maintaining the integrity of the computation.

A transversal gate applies operations independently and in parallel across corresponding qubits in different code blocks. Formally, a gate U is transversal if it can be written as the tensor product of single-qubit gates:

$$U = U_1 \otimes U_2 \otimes U_3 \ldots \otimes U_n,$$

where Ui acts on the ith physical qubit of a code block. If two blocks each encode a logical qubit across n physical qubits, a transversal implementation of a logical two-qubit gate, such as the logical CNOT, consists of applying CNOT gates independently between the corresponding physical qubits of the two blocks. The logical CNOT can be described as:

$$CNOT_L = CNOT_1 \otimes CNOT_2 \otimes CNOT_3 \ldots \otimes CNOT_n,$$

where each CNOT acts between the i-th qubit of the control block and the i-th qubit of the target block.

5.3. The Threshold Theorem

The Threshold Theorem is the central theoretical result behind the entirety of FTQC. It states that if the physical error rate per operation p is below a certain threshold pth, then arbitrarily long quantum computations can be performed reliably, with overhead that grows only polylogarithmically in the circuit size. Generally, pth is accepted to be around 10-2 to 10-4, with more specific ranges set depending on the user and use case. If p<pth, the quantum computations performed are still reliable, and the quantum computer is scalable. If the condition fails, then computations cannot be reliably performed due to a significant possibility of error.

The mechanism behind the Threshold Theorem can be understood through the concept of recursive encoding. Consider a quantum error-correcting code that can correct up to t errors in a block of physical qubits. If the physical error rate per operation is p, then after one level of encoding, the logical error rate is no longer linear in p. Instead, logical failure requires multiple physical faults within the same block, so the logical error rate scales approximately like pt+1, up to constant combinatorial factors. (Preskill, 1998)

This newly encoded logical qubit can then be encoded again using the same code, a process known as “concatenation”. In this second level, the previously obtained logical error rate becomes the effective “physical” error rate for the new block. As a result, the logical error rate is suppressed again by roughly the same power scaling law.

If the initial physical error rate is below a critical threshold value pthreshold, the recursive suppression continues with each level of encoding. The logical error rate would decrease rapidly with every level, enabling arbitrarily long computations to be performed reliably. If the physical error rate exceeds this threshold, however, recursive encoding no longer improves reliability and errors accumulate instead.

6. SIMULATION OF 3-QUBIT ERROR CORRECTION

6.1. Mathematical Proof of Error Correction in 3-Qubit System

Now, taking into consideration everything we have looked at so far, we will explore the propagation and correction of quantum errors within a 3-Qubit system, so that we can gain a better understanding of the tools used in building fault-tolerant quantum computers.

Let us begin this demonstration by defining a single logical qubit in the superposition:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \text{ where } \alpha, \beta \in \mathbb{C} \text{ and } |\alpha|^2 + |\beta|^2 = 1.$$

This qubit is encoded into 3 physical qubits as such:

$$|\psi\rangle_L = \alpha|000\rangle + \beta|111\rangle.$$

Let us suppose a bit-flip error, represented by the Pauli-X operator, acts on the second qubit due to external environmental conditions. The erroneous state then becomes:

$$|\psi\rangle_L = \alpha|010\rangle + \beta|101\rangle.$$

Here, the second qubit has been flipped, moving the state outside of the protected code space. In order to detect the error, we measure stabilizer generators S1=Z1Z2 and S2=Z2Z3. Measuring S1 yields an eigenvalue of -1, indicating that qubits 1 and 2 differ in value. Similarly, measuring S2 yields an eigenvalue of -1, indicating that qubits 2 and 3 differ in value. Therefore, the measured syndrome is:

$$(S_1, S_2) = (1, 1).$$

This identifies the error as a flip of qubit 2. We can then apply a corrective X gate to the second qubit.

$$X_2|\psi'\rangle = \alpha|000\rangle + \beta|111\rangle = |\psi\rangle_L.$$

The original state is thereby restored.

6.2. Computer Simulation of Error Correction in 3-Qubit System

Here is a simulation of a 3-Qubit bit-flip error, as well as the measurement and rectification of the error. This program was created using Qiskit, which is a Quantum Computing framework designed by IBM. The language utilised by this tool is

Python. The code for the program can be found in Appendix A. This simulation demonstrates to readers how syndrome measurement can identify and correct a single-bit flip error while preserving the encoded quantum information.

6.3. Single-Sampled Execution of Computer Simulation (Worked Example)

Qiskit uses zero-based indexing, and therefore, the simulation labels qubits as q_0 to q_4 rather than q_1 to q_5. Here is a single-sampled execution of the probabilistic noise model used to illustrate the mechanics of syndrome extraction and recovery. When run, this code will detail the state of the quantum system at every step throughout the process. The first useful result of the output would be the initialization of the logical qubit :

$$\text{Initializing logical qubit } |\psi\rangle = 0.70711|0\rangle + 0.70711|1\rangle$$

Upon the encoding of the logical qubit into three physical qubits, the circuit will be displayed as such:

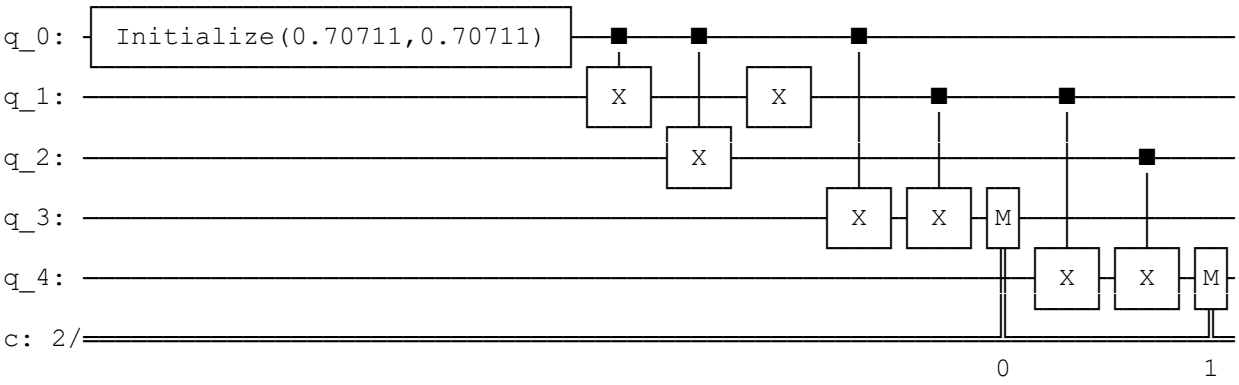
```
Circuit after encoding:
q_0: ┤ Initialize(0.70711,0.70711) ├──■──■──
q_1: ──────────────────────────────┤ X ├──┼──
q_2: ─────────────────────────────────┤ X ├──
q_3: ──────────────────────────────────────
q_4: ──────────────────────────────────────
c: 2/══════════════════════════════════════
```

Upon the introduction of a bit-flip error on qubit 1, the second qubit, the state of the circuit will change to:

```
> Injecting X error on random qubit...
> Circuit after error injection:
q_0: ┤ Initialize(0.70711,0.70711) ├──■──■──────
q_1: ──────────────────────────────┤ X ├──┼──┤ X ├
q_2: ─────────────────────────────────┤ X ├──────
q_3: ──────────────────────────────────────────
q_4: ──────────────────────────────────────────
c: 2/══════════════════════════════════════════
```

Syndrome extraction is then performed, and the Z1Z2 and Z2Z3 stabilizers are measured, displaying the circuit:

```
> Circuit after syndrome extraction:
```

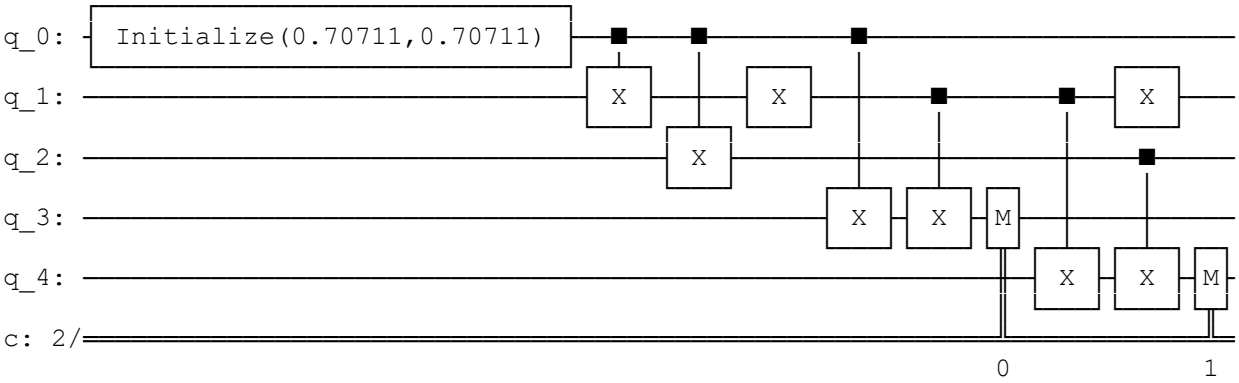


The syndrome measurement is then obtained, which allows the error correction method to be determined:

```
> Simulating syndrome measurement...
> Most common syndrome outcome: 11
> Detected X error on qubit 1 (q1). Applying correction...
```

The error correction technique is applied to the circuit, which is then printed as such:

```
> Full circuit after combining with correction:
```



This circuit shows the final circuit having been corrected, and the quantum information undisturbed.

6.4. Empirical Logical Error Rates Under a Probabilistic Bit-Flip Channel

While Section 6.3 presented a single sampled execution of the probabilistic noise model, we now examine the behaviour of the code across many trials. In this experiment, each of the three data qubits is subjected independently to a bit-flip error with probability p, corresponding to an i.i.d bit-flip channel.

For each trial, a logical state is prepared, encoded, subjected to stochastic noise, corrected using stabilizer measurements, and then decoded for measurement. A logical failure is an output measurement inconsistent with the originally prepared logical state. Repeating this process over many independent trials yields an empirical estimate of the logical error rate pL as a function of the physical bit flip probability p.

As expected, the three-qubit repetition code suppresses single-qubit bit-flip errors. We can see that logical failure events arise primarily when two or more physical bit flips occur within the same block, exceeding the code's correction capability. Consequently, for sufficiently small values of p, the observed logical error rate scales approximately quadratically with the physical error probability, reflecting the requirement of multiple simultaneous faults for logical failure. For small values of p, the dominant contribution to logical failure arises from two simultaneous bit-flip events, whose probability scales as $\binom{3}{2}p^2(1-p) \approx 3p^2$, while higher order terms are negligible.

Below, Table 2 summarises representative simulation (Appendix B) results over N trials for selected values of p. The empirical data confirm the anticipated reduction in logical error probability relative to the underlying physical error rate in the low-noise regime.

Table 2: Logical Error Rates of the Three-Qubit Repetition Code Under i.i.d. Bit-Flip Noise

Physical Error Rate, p	Trials, N	Logical Error Rate p_L	Theoretical p_L
0.01	10000	0.0003	0.000298
0.05	10000	0.0074	0.007375
0.10	10000	0.0275	0.028000
0.20	10000	0.104	0.104000

These results illustrate the central idea underlying fault tolerance in encoding, which is that logical failure requires correlated faults. In the low-noise regime, this leads to a nonlinear suppression of logical error rates relative to physical error rates, reflecting the redundancy built into the code.

7. RESULTS AND DISCUSSION

7.1. Reducing Overhead and Resource Costs

FTQC establishes the theoretical foundation for building reliable and scalable quantum computer systems in the presence of physical imperfections. However, realising FTQC in practice can yield significant challenges. One such challenge would be the immense overhead involved when implementing current error detection schemes. The encoding of a single logical qubit typically demands not just 3, but hundreds or thousands of physical qubits, especially when operating near the threshold error rates. (Fowler et al., 2012) Each logical operation also incurs significant amounts of circuit depth due to the

repeated cycles of error detection, syndrome extraction, and state preparation. (Devitt et al., 2013)

Reducing this overhead is an active field of research. Some of the common strategies include the design of more efficient quantum error-correcting codes, such as topological codes with low-weight stabilizers that reduce spatial and operational complexity. (Kitaev, 2003) On top of that, optimized concatenated code constructions that aim to minimize the total number of required qubits can also be implemented. (Knill, 2005) Recent advances in fault-tolerant circuit synthesis also focus on reducing the cost of non-Clifford gate constructions, which is a major bottleneck for scalable quantum computing. Achieving low-overhead fault-tolerant architectures is essential for realizing practical, large-scale quantum devices.

7.2. Development of Efficient Decoders

Another big challenge faced by FTQC is the development of scalable decoders. After syndrome extraction, the decoding algorithm must infer the likely error that occurred and prescribe the appropriate correction. For larger systems, decoding must be both fast enough to keep pace with hardware cycles and accurate enough to prevent logical errors. (Fowler, 2015)

Traditionally practiced decoding methods perform well under simple noise models but can struggle under correlated or biased noise. (Dennis et al., 2002) Emerging approaches leverage machine learning, including neural network-based decoders that can adapt to complex noise patterns and hardware-specific error profiles. (Torlai & Melko, 2017)

Other proposals investigate hardware-accelerated decoding, aiming to perform syndrome analysis and correction decision-making at cryogenic temperatures alongside the quantum processor. (Varsamopoulos et al., 2019)

Table 3: Comparison Of Decoder Approaches

Approach	Strengths	Weaknesses
Minimum Weight Perfect Matching	High accuracy near i.i.d. Pauli noise.	Classical compute can be heavy at scale, performance can also degrade under strong bias.
Union-Find	Very fast, low memory use, therefore scalable.	Slightly worse logical error rates than MWPM under comparable conditions.
Belief Propagation	Can incorporate biased or structured noise better than naïve Pauli-i.i.d. assumptions.	Requires tuning, may fail on loopy graphs.
Machine Learning Decoders	Can adapt to correlated, biased and hardware-specific noise, and inference can be fast after training	Requires good training data

There exist many such aforementioned approaches and proposals to scalable decoders. As shown in Table 3, decoder design is a three-way trade-off between accuracy, latency, and robustness to realistic noise. MWPM, otherwise known as Minimum Weight Perfect Matching, remains a strong accuracy baseline under near-i.i.d. Pauli error models, but its classical computational cost can become significant at scale. Faster methods, such as union-find and other greedy decoders, sacrifice some logical performance in exchange for real-time throughput. When the noise deviates from simple assumptions, probabilistic methods, and machine-learning decoders can better match the error structure, at the cost of model tuning, training data, and generalization issues.

Efficient decoders must balance computational complexity with error-correction performance, ensuring that decoding does not become the new bottleneck as quantum systems scale.

Importantly, decoder performance cannot be evaluated independently of the chosen code family and hardware platform. High-threshold codes such as surface codes are often paired with MWPM-style decoders, where accuracy is prioritized over simplicity. In contrast to this, subsystem and hardware-specialized codes may shift complexity from stabilizer measurement overhead to decoding logic. As system sizes grow, the interplay between code structure and decoding latency becomes a central architectural constraint, as a theoretically strong code may become impractical if the associated decoder cannot operate within hardware cycle times.

7.3. Exploring Alternative Error Models and Codes

Most early fault-tolerant schemes assume simplified noise models, such as independent and identically distributed (i.i.d.) bit-flip and phase-flip errors. However, actual quantum devices exhibit far more complicated error behaviors, including spatially and temporally correlated errors, non-Markovian effects, and leakage outside the computational subspace. (Preskill, 2018)

Table 4: Comparison Of Fault-Tolerant Protocols

Approach	Strengths	Weaknesses
Subsystem Codes	Reduced stabilizer measurement overhead, can tolerate certain correlated and measurement errors	More complex decoding
Bosonic Codes	Naturally match dominant physical noise	Require high-quality oscillators and control
Tailored Surface Code Variants	High fault-tolerance thresholds, adaptable to biased or correlated noise	Performance depends strongly on accurate noise characterization
Leakage-aware Protocols	Can prevent leakage from spreading and corrupting syndromes	Additional circuitry, time, and control complexity

To address these error behaviours, researchers are investigating alternative error models and novel quantum codes tailored to specific noise environments, as mentioned above (Table 4). Subsystem codes (Poulin, 2005), bosonic codes (Michael et al., 2016), and tailored surface code variants are some of the leading candidates for improving resilience against realistic hardware noise. At the same time, improved experimental noise characterization techniques are guiding the development of better theoretical models that more accurately capture error behavior. (Geller & Zhou, 2020) Codes and protocols designed with realistic noise in mind are expected to enhance the effectiveness of fault-tolerant architectures.

Beyond the idealized i.i.d. Pauli-noise setting, contemporary fault-tolerant design increasingly emphasizes protocols whose advantages and limitations are best understood in the context of realistic device noise. Subsystem codes are advantageous because their gauge structure can reduce stabilizer-measurement overhead and offer flexibility against certain measurement and correlated error mechanisms, although this typically comes at the cost of more intricate decoding and, in many instances, lower thresholds than leading topological codes. Bosonic codes, by contrast, exploit oscillator degrees of freedom to align the encoding with dominant physical processes such as loss and dephasing and can mitigate leakage by construction, but they rely on high-coherence modes and precise, platform-specific control. Tailored surface codes remain central due to their comparatively high thresholds and mature theoretical footing, yet they retain substantial resource demands, and their realized benefits depend sensitively on accurate noise modeling and characterization. Lastly, Leakage-aware protocols specifically address errors in which qubits leave the computational subspace by incorporating detection, suppression, or reset mechanisms, thereby preventing leaked states from persisting across cycles and contaminating syndrome extraction and logical operations.

These trade-offs reveal that threshold values alone do not determine architectural viability. Codes with high theoretical thresholds may incur issues such as substantial qubit overhead, increased stabilizer depth, or demanding classical decoding requirements. Conversely, codes optimized for specific physical noise processes may reduce overhead or measurement burden, but operate with tighter performance margins. Practically, a scalable fault-tolerant design emerges from balancing three competing objectives. Firstly, maximizing the threshold. Secondly, minimizing resource overhead. Finally, aligning with realistic hardware noise and control constraints. No existing code simultaneously optimizes all three, making architecture design inherently platform-dependent.

8. Conclusion

This paper has presented a basic overview of FTQC. Beginning with the framework for QEC, we demonstrated how codes such as the three-qubit bit-flip code serve as the first layer of defense against decoherence and physical noise. We then examined the necessity of fault tolerance in quantum computation. Fault tolerance, as we have observed, is not merely an enhancement to QEC but a rigorous discipline that ensures quantum algorithms remain functional even when components of the quantum circuit fail. By studying error models, syndrome extraction and correction, and the threshold theorem, we identified how carefully designed logical operations can prevent single-point failures from cascading across a system. (Terhal, 2015)

There are still practical challenges that exist in FTQC. High qubit overheads as well as slow and hardware-incompatible decoders hinder the deployment of FTQC on near-term devices. (Brown et al., 2016) However, emerging approaches

suggest a future where these limitations may be significantly mitigated, allowing for the construction of reliable large-scale quantum computer systems.

As a concrete next step, readers can examine surface-code implementations adapted to the constraints of current hardware platforms, and how syndrome extraction is paired with practical decoding strategies. This provides a route from the abstract principles of fault tolerance to real engineering considerations that need to be taken.

As quantum technologies advance beyond the NISQ era, FTQC will be indispensable. It not only offers a path to preserving quantum information over long durations, but also ensures that the benefits of quantum computation can be scaled up safely and reliably. In this light, fault tolerance is the mechanism by which the promise of quantum computing becomes physically and practically realisable.

9. References

Aharonov, D., & Ben-Or, M. (2008). Fault-tolerant quantum computation with constant error rate. SIAM Journal on Computing, 38(4), 1207–1282. https://doi.org/10.1137/S0097539799359385

Bravyi, S., & Kitaev, A. (2005). Universal quantum computation with ideal Clifford gates and noisy ancillas. Physical Review A, 71(2), 022316. https://doi.org/10.1103/PhysRevA.71.022316

Dennis, E., Kitaev, A., Landahl, A., & Preskill, J. (2002). Topological quantum memory. Journal of Mathematical Physics, 43(9), 4452–4505. https://doi.org/10.1063/1.1499754

Devitt, S. J., Munro, W. J., & Nemoto, K. (2013). Quantum error correction for beginners. Reports on Progress in Physics, 76(7), 076001. https://doi.org/10.1088/0034-4885/76/7/076001

Fowler, A. G. (2015). Minimum weight perfect matching of fault-tolerant topological quantum error correction in average O(1) parallel time. Quantum Information & Computation, 15(1–2), 145–158. http://www.rintonpress.com/xxqic15/qic-15-12/0145-0158.pdf

Fowler, A. G., Mariantoni, M., Martinis, J. M., & Cleland, A. N. (2012). Surface codes: Towards practical large-scale quantum computation. Physical Review A, 86(3), 032324. https://doi.org/10.1103/PhysRevA.86.032324

Geller, M. R., & Zhou, Z. (2020). Efficient error models for fault-tolerant architectures and the Pauli twirling approximation. Physical Review Research, 2(1), 013073. https://doi.org/10.1103/PhysRevResearch.2.013073

Gottesman, D. (1997). Stabilizer codes and quantum error correction (Doctoral dissertation, California Institute of Technology). California Institute of Technology. https://arxiv.org/abs/quant-ph/9705052

Gottesman, D. (1998). Theory of fault-tolerant quantum computation. Physical Review A, 57(1), 127–137. https://doi.org/10.1103/PhysRevA.57.127

Kitaev, A. Y. (2003). Fault-tolerant quantum computation by anyons. Annals of Physics, 303(1), 2–30. https://doi.org/10.1016/S0003-4916(02)00018-0

Knill, E. (2005). Quantum computing with realistically noisy devices. Nature, 434(7029), 39–44. https://doi.org/10.1038/nature03350

Michael, M. H., Silveri, M., Brierley, R. T., Albert, V. V., Salmilehto, J., Jiang, L., & Girvin, S. M. (2016). New class of quantum error-correcting codes for a bosonic mode. Physical Review X, 6(3), 031006. https://doi.org/10.1103/PhysRevX.6.031006

Nielsen, M. A., & Chuang, I. L. (2010). Quantum computation and quantum information (10th anniversary ed.). Cambridge University Press.

Poulin, D. (2005). Stabilizer formalism for operator quantum error correction. Physical Review Letters, 95(23), 230504. https://doi.org/10.1103/PhysRevLett.95.230504

Preskill, J. (1998). Reliable quantum computers. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 454(1969), 385–410. https://doi.org/10.1098/rspa.1998.0167

Preskill, J. (2018). Quantum computing in the NISQ era and beyond. Quantum, 2, 79. https://doi.org/10.22331/q-2018-08-06-79

Reichardt, B. W. (2005). Quantum universality from magic states distillation applied to CSS codes. Quantum Information Processing, 4(3), 251–264. https://doi.org/10.1007/s11128-005-7499-4

Shor, P. W. (1995). Scheme for reducing decoherence in quantum computer memory. Physical Review A, 52(4), R2493–R2496. https://doi.org/10.1103/PhysRevA.52.R2493

Torlai, G., & Melko, R. G. (2017). Neural decoder for topological codes. Physical Review Letters, 119(3), 030501. https://doi.org/10.1103/PhysRevLett.119.030501

Varsamopoulos, S., Criger, B., & Bertels, K. (2019). Decoding small surface codes with feedforward neural networks. Quantum Science and Technology, 3(1), 015004. https://doi.org/10.1088/2058-9565/aab822

Terhal, B. M. (2015). Quantum error correction for quantum memories. Reviews of Modern Physics, 87(2), 307–346. https://doi.org/10.1103/RevModPhys.87.307

Brown, B. J., Loss, D., Pachos, J. K., Self, C. N., & Wootton, J. R. (2016). Quantum memories at finite temperature. Reviews of Modern Physics, 88(4), 045005. https://doi.org/10.1103/RevModPhys.88.045005

Acknowledgements

I would like to thank Dr. Adam Almakroudi for his guidance in understanding and developing the mathematical foundations of this paper. I am also grateful to the Centre for Quantum Technologies for fostering a community of like-minded, research-driven students who deepened my understanding of quantum computing and its academic literature.

I extend my thanks to Jared Cheang, a former teacher of mine, who first ignited my fascination with quantum physics and inspired the direction of this work. I am also grateful to Veniko Borislavov Belinski for being a trusted colleague with whom I

could exchange ideas and discuss concepts throughout the research process. Finally, I would like to thank Triparna Poddar for her support and encouragement throughout what was often a demanding and hectic research journey.

Author Biography

Raghav Sriram is an independent researcher focusing on hardware security, electronic engineering, and scalable quantum computing. His work is interdisciplinary in nature, bridging research perspectives across different technical domains.

He is the author of "An Overview of Assuring Fault Tolerance in Quantum Computers Through Quantum Error Correction," a structured review examining stabiliser codes, surface codes, subsystem codes, and magic state distillation within a unified mathematical framework. His research integrates formal quantum theory (including linear algebraic representations of quantum states, Kraus operator formulations of noise, and threshold theorems) with simulation-based validation using Qiskit.

Beyond quantum information science, Raghav conducts applied research in hardware security. His work focuses on reverse engineering, side-channel analysis, and embedded system security. He is particularly interested in the emerging field of quantum computer security and the protection of next-generation computing architectures.

Raghav is going to pursue his university education in Electronic Engineering in the United Kingdom. He aims to contribute meaningfully to the development of resilient, high-performance computing systems and produce work that advances both theoretical understanding and real-world technological capability.

Mentor Contribution Statement

Dr. Adam Almakroudi served in an advisory and educational capacity, acting as a mentor during the development of this paper. His role consisted of providing conceptual guidance in the author's study of the mathematical framework underlying quantum computing and quantum error correction, including algebraic formalisms, quantum states, and quantum gate operations.

He assisted in refining the scope of the paper by providing vital feedback on the topics selected and the appropriate breadth and depth for the intended audience. This guidance helped ensure that the paper was well-scoped, properly directed, and academically coherent.

10. APPENDIX A

Qiskit 3-Qubit Bit Flip and Correction Simulation Code:

```
from qiskit import QuantumCircuit, transpile
from qiskit_aer import Aer
from qiskit.quantum_info import Statevector
import numpy as np

# --- Step 1: Initialize logical qubit ---
```

```
alpha = 1 / np.sqrt(2)
beta = 1 / np.sqrt(2)
print("> Initializing logical qubit |ψ⟩ = {:.5f}|0⟩ + {:.5f}|1⟩".format(alpha, beta))
# 3 data qubits (q0, q1, q2), 2 ancilla qubits (q3, q4), 2 classical bits (c0, c1)
qc = QuantumCircuit(5, 2)

# --- Step 2: Encode the logical qubit into the 3-qubit repetition code ---
qc.initialize([alpha, beta], 0)  # logical qubit on q0
qc.cx(0, 1)
qc.cx(0, 2)
print("\n> Circuit after encoding:")
print(qc.draw('text'))

# --- Step 3: Inject a bit-flip (X) error on one of the qubits ---
print("\n> Injecting X error on random qubit...")
p = 0.2 # physical bit-flip probability
for q in [0, 1, 2]:
      if np.random.rand() < p:
            qc.x(q)
print("\n> Circuit after error injection:")
print(qc.draw('text'))

# --- Step 4: Syndrome extraction using ancilla qubits (q3, q4) ---
# Measure Z0Z1 using q3
qc.cx(0, 3)
qc.cx(1, 3)
qc.measure(3, 0)
# Measure Z1Z2 using q4
qc.cx(1, 4)
qc.cx(2, 4)
qc.measure(4, 1)
print("\n> Circuit after syndrome extraction:")
print(qc.draw('text'))

# --- Step 5: Simulate the circuit and obtain syndrome ---
print("\n> Simulating syndrome measurement...")
backend = Aer.get_backend('qasm_simulator')
job = backend.run(transpile(qc, backend), shots=1024)
result = job.result()
counts = result.get_counts()
syndrome = max(counts, key=counts.get)
print("> Most common syndrome outcome:", syndrome)

# --- Step 6: Determine and apply correction based on syndrome ---
correction = QuantumCircuit(5)
if syndrome == '10':
    print("> Detected X error on qubit 0 (q0). Applying correction...")
    correction.x(0)
```

```
elif syndrome == '11':
    print("> Detected X error on qubit 1 (q1). Applying correction...")
    correction.x(1)
elif syndrome == '01':
    print("> Detected X error on qubit 2 (q2). Applying correction...")
    correction.x(2)
else: print("> No error detected. No correction applied.")
print("\n> Correction circuit:")
print(correction.draw('text'))

# --- Step 7: Combine full circuit with correction ---
full_circuit = qc.compose(correction)
print("\n> Full circuit after combining with correction:")
print(full_circuit.draw('text'))

# --- Step 8: Simulate final statevector after correction ---
print("\n> Simulating final corrected statevector...")
state_backend = Aer.get_backend('statevector_simulator')
final_job = state_backend.run(transpile(full_circuit, state_backend))
final_result = final_job.result()
final_statevector = final_result.get_statevector()
print("\n> Final corrected statevector:")
print(final_statevector)
print("\n> Simulation complete. Logical qubit successfully protected and corrected.")
```

11. APPENDIX B

Qiskit 3-Qubit Bit Flip and Correction Logical Error Rate Tabulation Code:

```
from qiskit import QuantumCircuit, transpile
from qiskit_aer import Aer
import numpy as np

backend = Aer.get_backend("qasm_simulator")

def build_trial_circuit(prep_bit: int, p: float, rng: np.random.Generator) ->
QuantumCircuit:

    qc = QuantumCircuit(5, 3)

    # 1) Prepare |0> or |1> on q0
    if prep_bit == 1:
        qc.x(0)

    # 2) Encode the 3-qubit repetition code
    qc.cx(0, 1)
    qc.cx(0, 2)
```

```
    # 3) i.i.d. bit-flip channel on the data qubits
    for q in [0, 1, 2]:
        if rng.random() < p:
            qc.x(q)

    # 4) Syndrome extraction
    # Measure Z0Z1 using ancilla q3 -> c0
    qc.cx(0, 3)
    qc.cx(1, 3)
    qc.measure(3, 0)

    # Measure Z1Z2 using ancilla q4 -> c1
    qc.cx(1, 4)
    qc.cx(2, 4)
    qc.measure(4, 1)

    # --- Minimal fix: apply recovery in-circuit using measured syndrome (c1 c0) ---
    # Classical register is 3 bits (c0, c1, c2). At this point c2 is still 0.
    # Values for (c1 c0): 00->0, 01->1, 10->2, 11->3
    qc.x(0).c_if(qc.cregs[0], 2)  # 10 -> X on q0
    qc.x(1).c_if(qc.cregs[0], 3)  # 11 -> X on q1
    qc.x(2).c_if(qc.cregs[0], 1)  # 01 -> X on q2
    # ------------------------------------------------------------------------

    # 5) Decode (inverse of encoding)
    qc.cx(0, 2)
    qc.cx(0, 1)

    # 6) Measure logical output q0 -> c2
    qc.measure(0, 2)

    return qc

def apply_correction_postprocess(outcome: str) -> int:
    """
    Apply the correction logic
    Syndrome Mapping (c1 c0):
      00 -> no correction
      10 -> X on q0
      11 -> X on q1
      01 -> X on q2

    Since we decode before measuring q0, only the '10' case (X on q0) flips the decoded
logical bit.
    The outcome is a 3-bit string: c2 c1 c0 (MSB -> LSB).
    """
    c2 = int(outcome[0])
    c1 = int(outcome[1])
```

```
    c0 = int(outcome[2])

    # Recovery is applied in-circuit; return decoded logical bit directly.
    return c2

def one_trial_failure(prep_bit: int, p: float, rng: np.random.Generator) -> bool:
    """
    Run one trial and return True if logical failure, otherwise return False.
    """
    qc = build_trial_circuit(prep_bit, p, rng)
    result = backend.run(transpile(qc, backend), shots=1).result()
    outcome = next(iter(result.get_counts().keys()))
    logical_out = apply_correction_postprocess(outcome)
    return logical_out != prep_bit

def estimate_logical_error_rate(p: float, N: int = 10000, seed: int = 42) -> float:
    """
    Estimate the logical error rate p_L over N independent trials.
    """
    rng = np.random.default_rng(seed)
    failures = 0
    for i in range(N):
        prep_bit = i % 2
        failures += one_trial_failure(prep_bit, p, rng)
    return failures / N

def theoretical_pL(p: float) -> float:
    """
    For i.i.d. bit flips on 3 qubits, the code fails when over 1 flip occurs:
      p_L = P(2 flips) + P(3 flips) = 3 p^2 (1-p) + p^3
    """
    return 3*(p**2)*(1 - p) + (p**3)

if __name__ == "__main__":
    # Here, please choose the N and p values for Table 2
    N_trials = 5000
    ps = [0.01, 0.05, 0.10, 0.20]

    print("p\tN\tp_L (observed)\tp_L (theoretical)")
    for p in ps:
        pL_obs = estimate_logical_error_rate(p=p, N=N_trials, seed=7)
        pL_th = theoretical_pL(p)
        print(f"{p:.2f}\t{N_trials}\t{pL_obs:.6f}\t\t{pL_th:.6f}")
```