Quantum Error Correction - Theory and Hands-on

Jérôme Lenssen, Delphine Martres, Hemanadhan Myneni, Franz G. Fuchs

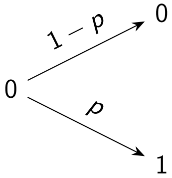
9:00 - 12:00, 3. December 2024

Table of Contents

Outline

Classical noisy channel

- Send k-bit message across a noisy channel.
- Channel flips one bit independently with low probability p



- How do you protect from the noise?

Repetition code

- Repeat information, so e.g., send $x = 000$ instead of $x = 0$

$$x \xrightarrow{\text{noise}} \tilde{x}$$

- one receives:
  - 000 with probability $(1-p)^3$,
  - 100, 010, 001 each with probability $(1-p)^2 p$,
  - 011, 101, 110 each with probability $(1-p)p^2$, and
  - 111 with probability p^3.
- Let's say we receive $\tilde{x} = 010$
- Assuming at most one error occurred, we can take a majority vote to decode $x = 000$

Error correction process

$$x \xrightarrow{\text{encoding}} y \xrightarrow{\text{noise}} \tilde{y} \xrightarrow{\text{decoder}} \tilde{x}$$

- $\mathcal{E}(x) = y$ encodes a k bit message x, into an n bits
- A **codeword** is an element of the image of $\mathcal{E}$:
  Set of all codewords $C = \text{Im}(\mathcal{E})$
- E.g., 3-repetition code 0, $k = 1, n = 3$, we have $C = \{000, 111\}$
- If one or two bits are flipped, the error is **detectable**
- If all bits are flipped, the error is **undetectable** = **logical error**

Simple Parity-Check Code

Encoding: Given a 3-bit message (a, b, c), the parity-check code encodes it as:

$$E(a, b, c) = (a, b, c, z) \quad \text{where} \quad z = (a + b + c) \bmod 2$$

Properties:

- z indicates whether the sum of a, b, c is even $(z = 0)$ or odd $(z = 1)$.
- Any single-bit error can be detected:
  - If a, b, or c is flipped, $z \neq (a + b + c) \bmod 2$.
  - If z is flipped, it no longer corresponds to the parity of a, b, c.

Limitation: Errors cannot be corrected.

Hamming Codes - The Idea

- Hamming introduced a method to **correct errors** using parity-check bits.
- Example: A 4-bit message $x = (a, b, c, d)$ with 3 parity-check bits:

$$z_1 = a + b + d, \quad z_2 = a + c + d, \quad z_3 = b + c + d \pmod 2$$

- Encoded message:

$$y = E(a, b, c, d) = (a, b, c, d, z_1, z_2, z_3)$$

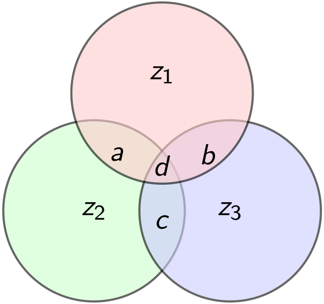
Parity-check Visualization

Error Detection:

- If a single bit is flipped, certain parity-checks will fail.
- Example:
  - If a is flipped, z_1 and z_2 will fail, while z_3 remains valid.

Error Correction:

- The pattern of failed parity-checks indicates the position of the flipped bit.
- Example: Flip in d causes all z_1, z_2, z_3 to fail.



Introduction to Linear Codes

- Linear codes generalize the concept of transmitting a message with parity-check bits.
- A **linear code** uses a matrix $\boldsymbol{G}$—called the **generator matrix**—to encode the message:

$$\boldsymbol{y} = \boldsymbol{G}\boldsymbol{x}.$$

- The message $\boldsymbol{x}$ has length k, and is supplemented with m parity-check bits such that the encoded message $\boldsymbol{y}$ has length $n = k + m$.
- The generator matrix $\boldsymbol{G}$ can be written as:

$$\boldsymbol{G} = \begin{pmatrix} I_k \\ \hline \boldsymbol{A} \end{pmatrix}$$

  - I_k: $k \times k$ identity matrix (reproduces the message bits),
  - $\boldsymbol{A}$: $m \times k$ matrix (defines parity-check operations).

Example: Generator Matrix of the Hamming Code

For the **[7,4]-Hamming code**, the generator matrix is:

$$\boldsymbol{G} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

Encoding a message $\boldsymbol{x} = (a, b, c, d)^T$:

$$\boldsymbol{Gx} = \begin{pmatrix} a \\ b \\ c \\ d \\ a+b+d \\ a+c+d \\ b+c+d \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \\ d \\ z_1 \\ z_2 \\ z_3 \end{pmatrix}.$$

Properties of Generator Matrices

Code is defined as image of G:

1. The codewords are the set of all linear combinations of the columns of $\boldsymbol{G}$.
2. To find all the codewords, just calculate all the $\boldsymbol{y}$ of the form $\boldsymbol{y} = a_1\boldsymbol{g_1} + \cdots + a_k\boldsymbol{g_k}$, where $\boldsymbol{g_i}$ is the i^{th} column of G and $a_1, \ldots, a_k \in \{0, 1\}$.
3. Elementary row and column operations on $\boldsymbol{G}$ do not change the code.
4. Using Gaussian elimination, $\boldsymbol{G}$ can always be transformed into the standard form:

$$\boldsymbol{G} = \begin{pmatrix} I_k \\ \hline \boldsymbol{A} \end{pmatrix}.$$

Parity-Check Matrix

An equivalent representation of a linear code is the **parity-check matrix $\boldsymbol{H}$**:

$$\boldsymbol{H}\boldsymbol{y} = \mathbf{0},$$

- $\boldsymbol{H}$ is an $m \times n$ matrix,
- $\boldsymbol{y}$ is a codeword if and only if $\boldsymbol{H}\boldsymbol{y} = \mathbf{0}$.
- set of all codewords $C = \mathrm{Ker}(H)$

For the [7,4]-Hamming code:

$$\boldsymbol{H} = \left(\begin{array}{cccc|ccc} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array} \right).$$

- A received word $\widetilde{\boldsymbol{y}}$ can be checked for errors by evaluating $\boldsymbol{H}\widetilde{\boldsymbol{y}}$.
- Errors can often be located and corrected using this method.

Generator Matrix and Parity-Check Matrix

For a generator matrix of the form:

$$\boldsymbol{G} = \left(\frac{I_k}{\boldsymbol{A}}\right)$$

the corresponding parity-check matrix can be written as:

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{A} & I_m \end{pmatrix}.$$

Why Use the Parity-Check Matrix?

- A received vector $\widetilde{\boldsymbol{y}} = \boldsymbol{y} + \boldsymbol{e}$ combines the original codeword $\boldsymbol{y}$ and an error vector $\boldsymbol{e}$.
- Applying the parity-check matrix $\boldsymbol{H}$ gives:

$$\boldsymbol{H}\widetilde{\boldsymbol{y}} = \boldsymbol{H}(\boldsymbol{y} + \boldsymbol{e}) = \boldsymbol{H}\boldsymbol{e}.$$

- The result $\boldsymbol{s} = \boldsymbol{H}\boldsymbol{e}$ is known as the **syndrome**.
- The syndrome identifies errors by revealing violated parity-check equations: $s_i = 1$ indicates a violation.

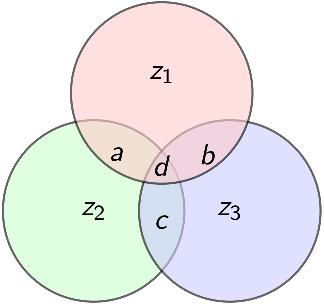
Decoding:

- Decoding finds the most probable error $\boldsymbol{e}$ that explains the syndrome $\boldsymbol{s}$.
- A violated parity-check equation points to specific bits involved in the error.

Syndrome Table for Error Correction

The following table shows the bit we choose to correct for each of the 8 possible syndromes

Syndrome	000	100	010	001	110	101	011	111
Correction	$\emptyset$	z_1	z_2	z_3	a	b	c	d



In quantum error correction, the syndrome can/has to be measured without disturbing the quantum state

Decoding linear codes

Decoding consists of finding the original message given its noisy encoded version.

- There are 2^n possible syndromes.
- We define an **efficient decoder** as an algorithm that accomplishes this task in polynomial time in n.

Given the parity-check matrix $\boldsymbol{H}$. Let's assume errors follow a certain distribution $P(\boldsymbol{e})$.

- Given the received syndrome s, we want to find the most likely error e.
- The **goal of an ideal decoder**:

  Find the vector $\boldsymbol{e}$ that **maximizes** the probability $P(\boldsymbol{e} \mid \boldsymbol{s})$.

Applying Bayes' Rule

Using Bayes' rule, we can write:

$$P(\boldsymbol{e} \mid \boldsymbol{s}) = \frac{P(\boldsymbol{s} \mid \boldsymbol{e})P(\boldsymbol{e})}{P(\boldsymbol{s})}.$$

- $P(\boldsymbol{s})$ does not depend explicitly on $\boldsymbol{e}$, and can be ignore for solving the maximization problem over $\boldsymbol{e}$
- Any valid error $\boldsymbol{e}$ must satisfy $\boldsymbol{He} = \boldsymbol{s}$, so:

$$P(\boldsymbol{s} \mid \boldsymbol{e}) = \begin{cases} 1, & \text{if } \boldsymbol{He} = \boldsymbol{s} \\ 0, & \text{otherwise.} \end{cases}$$

Our optimization problem then becomes:

$$\max_{\boldsymbol{e} \in \{0,1\}^n} P(\boldsymbol{e}) \quad \text{subject to} \quad \boldsymbol{He} = \boldsymbol{s}.$$

Special Case: Independent Errors

- In the case where errors are iid, we have:

$$P(\boldsymbol{e}) = \prod_{i=1}^{n} P(e_i).$$

- Let $P(e_i = 1) = p$ and $P(e_i = 0) = 1 - p$. Then:

$$P(\boldsymbol{e}) = p^{|\boldsymbol{e}|}(1-p)^{n-|\boldsymbol{e}|},$$

  where $|\boldsymbol{e}|$ is the Hamming weight of $\boldsymbol{e}$.
- For $p < 0.5$, the probability $P(\boldsymbol{e})$ increases when the weight of $\boldsymbol{e}$ decreases. Therefore, our optimization problem reduces to finding the error of minimum weight that satisfies $\boldsymbol{He} = \boldsymbol{s}$:

$$\min_{\boldsymbol{e} \in \{0,1\}^n} |\boldsymbol{e}| \quad \text{subject to} \quad \boldsymbol{He} = \boldsymbol{s}.$$

Maximum A Posteriori (MAP) Decoding and Its Challenges

MAP Decoder: Any decoder that explicitly solves

$$\max_{\boldsymbol{e}\in\{0,1\}^n} P(\boldsymbol{e} \mid \boldsymbol{s}),$$

is called MAP decoder, and is considered an *ideal decoder*.

Challenges:

- A naive approach requires searching all 2^n possible error vectors, leading to exponential time complexity.
- The MAP decoding problem is **NP-complete**, meaning no general polynomial-time algorithm is likely to exist.

Efficient Decoding for Special Codes:

- Certain structured codes (e.g., Hamming codes, repetition codes) allow polynomial-time decoding.

Heuristic Approaches to MAP Decoding

Heuristics: Approximate solutions for MAP decoding that are efficient and perform well in practice.

Belief Propagation Algorithm:

- An iterative, linear-time algorithm.
- Exploits the factorization of $P(\boldsymbol{e} \mid \boldsymbol{s})$ over a graph (e.g., Tanner graph).
- Widely used in classical error-correction and also in quantum error correction.

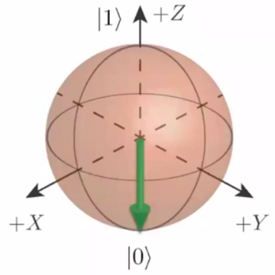
Outline

Quantum Logic

- Quantum bit/Qubit

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

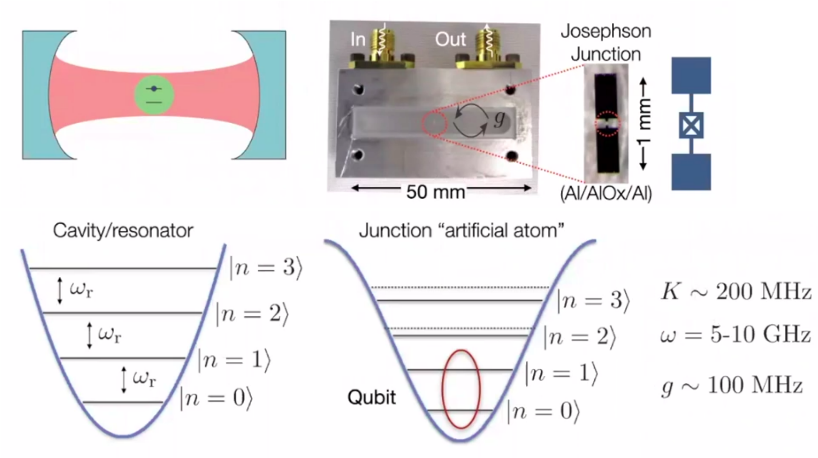
$$|\alpha|^2 + |\beta|^2 = 1$$



- Universal logical operations, gates, unitaries: Hadamard, S-gate, T-gate, CNOT

- **Measurements:**
  - Set of operators $\{M_i\}$ such that $\sum_i M_i^\dagger M_i = I$
  - Probability of outcome i is $p(i) = \langle\psi|M_i^\dagger M_i|\psi\rangle$
  - State after obtaining outcome i is $\frac{M_i|\psi\rangle}{\sqrt{p(i)}}$

Hardware: Superconducting Circuits

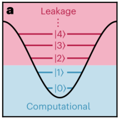


$$\hat{H} = \omega_r \hat{a}^\dagger \hat{a} \qquad \hat{H} \sim \omega_q \hat{b}^\dagger \hat{b} - K \hat{b}^{\dagger 2} \hat{b}^2$$

PRA 69, 062320 (2004)

Sources of Quantum Noise / Errors

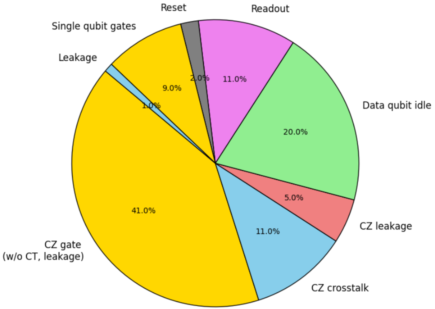
- **Decoherence:**
  - T1 relaxation: Energy decay from the $|1\rangle \to |0\rangle$
  - T2 dephasing: Loss of phase coherence in superposition states.
- **Gate Errors:** Imperfect implementation of quantum gate operations, leading to inaccuracies.
- **Measurement Errors:** Errors during the readout of qubit states, resulting in incorrect outputs.
- **Cross-Talk:** Interference between neighboring qubits during operations, reducing fidelity.
- **Leakage Errors:** Qubits transitioning to higher energy states outside the computational basis.

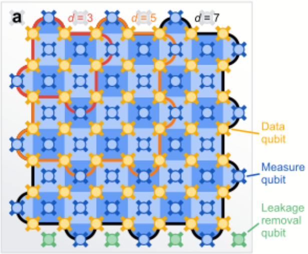


- **Stray Interactions:** Unintended couplings during gate operations.
- **Idle Errors:** Errors occurring while qubits remain idle due to environmental interactions.
- **External Noise:** Electromagnetic interference or cosmic rays.

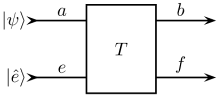
Error budget distribution

distance-5 surface code on 72-qubit processor (arXiv:2408.13687v1)





Noisy Quantum Channels



- A noisy quantum channel introduces errors during transmission or processing.
- Examples of noise effects:
  - Degradation of quantum states.
  - Reduction in entanglement and coherence.
  - Significant impact on fidelity and performance.
- Kraus operators provide a powerful framework to describe and analyze noise in quantum systems.

Kraus Operators

- Describe the evolution of quantum states in open systems.
- Evolution of a density matrix (ρ) under Kraus operators:

$$\rho' = \sum_i K_i \rho K_i^\dagger$$

- Completeness relation ensures trace preservation:

$$\sum_i K_i^\dagger K_i = I$$

- These operators model common errors such as bit-flip, phase-flip, and depolarization.

Bit-Flip Channel

- Models noise where qubits flip between $|0\rangle$ and $|1\rangle$ with probability p.
- Quantum state evolution:

$$\mathcal{T}(\rho) = (1 - p)\rho + pX\rho X^\dagger$$

- Kraus operators:

$$K_0 = \sqrt{1 - p}\, I, \quad K_1 = \sqrt{p}\, X$$

Depolarizing Channel

- Randomizes the qubit state with probability p.
- Channel action:

$$\mathcal{E}(\rho) = (1-p)\rho + \frac{p}{3}(X\rho X^\dagger + Y\rho Y^\dagger + Z\rho Z^\dagger)$$

- Kraus operators:

$$K_0 = \sqrt{1-p}\, I, \quad K_1 = \sqrt{\frac{p}{3}}\, X, \quad K_2 = \sqrt{\frac{p}{3}}\, Y, \quad K_3 = \sqrt{\frac{p}{3}}\, Z$$

Amplitude Damping Channel

- Models energy dissipation, such as photon loss.
- Channel action:

$$\mathcal{E}(\rho) = E_0 \rho E_0^\dagger + E_1 \rho E_1^\dagger$$

- Kraus operators:

$$E_0 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{bmatrix}, \quad E_1 = \begin{bmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{bmatrix}$$

Quantum Error Correction

- Quantum error correction (QEC) is essential for protecting quantum information against noise and decoherence.
- The primary goals of QEC are:
  - Detect errors without disturbing the quantum information.
  - Correct errors to restore the original quantum state.
  - Ensure fault-tolerant quantum computation.

Errors in quantum computers

- Classically, bits can flip (or be erased).
  i.e., $0 \to 1$ and $1 \to 0$ with some probability p.
- Qubits have a larger state space, so more things can go wrong.
  - Any operation that can be considered a gate can also introduce an error.
  - Examples include Pauli errors (X, Z, Y).

$X\|0\rangle = \|1\rangle$ $X\|1\rangle = \|0\rangle$ **Bit flip**	$Z\|0\rangle = \|0\rangle$ $Z\|1\rangle = -\|1\rangle$ **Phase flip**	$Y\|0\rangle = i\|1\rangle = iXZ\|0\rangle$ $Y\|1\rangle = -i\|0\rangle = iXZ\|1\rangle$ **Bit & phase flip**

The Most Important Fact About QEC

- Errors are inherently continuous (analog). How can we hope to correct these?
- Suppose some error E introduces a relative phase:

$$E|\psi\rangle = \alpha|0\rangle + e^{i\delta}\beta|1\rangle$$

- The angle δ could (in principle) be infinitesimal.
- Any error can be written as discrete Pauli errors with continuous coefficients:
  - This is because the Pauli matrices (+ the Identity) span $\mathbb{C}^{2\times 2}$.
  - For any E and ψ:

$$E|\psi\rangle = (e_0 I + e_1 X + e_2 Y + e_3 Z)|\psi\rangle$$

- But the coefficients e_i could still be infinitesimal, in principle.

The Most Important Fact About QEC

- Measurement **turns** continuous errors into discrete errors.
- Suppose we measure the error state using operators $\{M_i\}$:

$$E|\psi\rangle = (e_0 I + e_1 X + e_2 Y + e_3 Z)|\psi\rangle$$

- Then, with probability $p(i)$, the state collapses to: $\frac{M_i E|\psi\rangle}{\sqrt{p(i)}}$
- This process collapses the superposition and reduces the continuous coefficients to a global phase, which is irrelevant.
  - For example, we could choose M_i such that: $E|\psi\rangle \to \eta_i \sigma_i |\psi\rangle$
  - Here, $\sigma_i \in \{I, X, Y, Z\}$ is a discrete error that can be corrected.
  - The coefficient $\eta_i \in \mathbb{C}$ is continuous but represents a global phase and hence does not matter.

Outline

Classical error correction: The repetition code

- A key concept in error correction is adding **redundancy**.
- For example, given a bit, we can make three copies of it:
  - $0 \to 000, \quad 1 \to 111$
  - This is known as the (classical) **repetition code**.
  - The idea is very simple: If an error occurs on one bit only, we can correct it by looking at the other two bits and taking a majority vote.
- Given the classical repetition code, we might try to do the same with qubits, i.e. map

$$|\psi\rangle \to |\psi\rangle|\psi\rangle|\psi\rangle$$

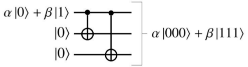
- This is not possible due to the **"no cloning theorem"**

QEC: Can We Add Any Redundancy?

- From the no-cloning theorem, we know it is **not** possible to make exact copies of a quantum state as in the classical repetition code.
- Can we copy information?
- **Claim:** We can "copy basis information" in the following sense:

$$\alpha|0\rangle + \beta|1\rangle \rightarrow \alpha|000\rangle + \beta|111\rangle$$

- Note that this encoding circuit entangles the "input" qubit with two other qubits.



- Errors in quantum computers are often caused by qubits entangling with their **environment**.

Repetition code for bit flip errors

- The encoding $\alpha|0\rangle + \beta|1\rangle \rightarrow \alpha|000\rangle + \beta|111\rangle$ gives us **redundancy**. Now what?
- We need to check which errors (if any) occured in the encoded state.
- We do this by (projective) measurements. What projections should we apply to find out what happened?
- There are four possible things that can happen:

No qubit was flipped.	$P_0 = |000\rangle\langle000| + |111\rangle\langle111|$
The first qubit was flipped.	$P_1 = |100\rangle\langle100| + |011\rangle\langle011|$
The second qubit was flipped.	$P_2 = |010\rangle\langle010| + |101\rangle\langle101|$
The third qubit was flipped.	$P_3 = |001\rangle\langle001| + |110\rangle\langle110|$

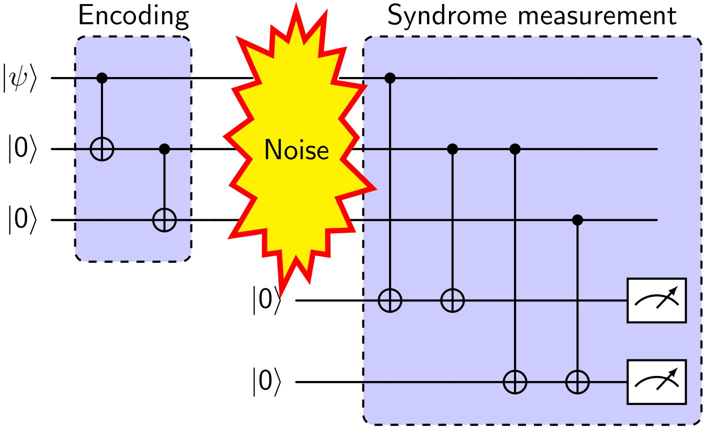
Turning the table

- By measuring these operators, we learn what errors (if any) occurred.
- Since we know which error occurred, we can **correct** it.

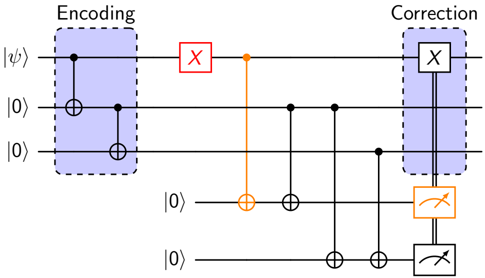
Syndrome measurement	Meaning	Correction operator
$P_0 = \|000\rangle\langle 000\| + \|111\rangle\langle 111\|$	No qubit was flipped.	I
$P_1 = \|100\rangle\langle 100\| + \|011\rangle\langle 011\|$	The first qubit was flipped.	X_0
$P_2 = \|010\rangle\langle 010\| + \|101\rangle\langle 101\|$	The second qubit was flipped.	X_1
$P_3 = \|001\rangle\langle 001\| + \|110\rangle\langle 110\|$	The third qubit was flipped.	X_2

- But since measurement **collapses** the state, we need to use ancilla qubits for syndrome measurement.

Bitflip Repetition Code: circuit implementation

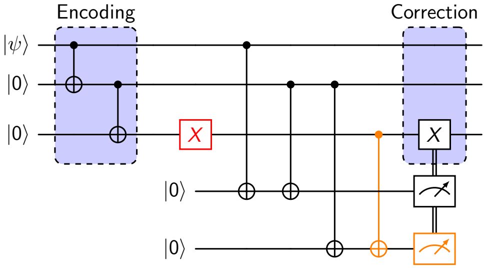


Bitflip Repetition Code: circuit implementation



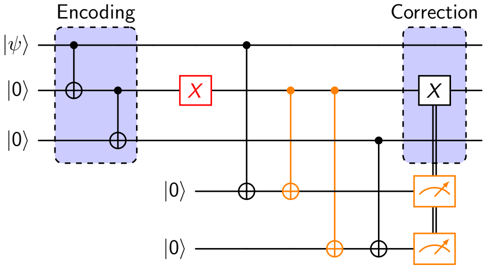
Syndrome	Correction
00	I
10	X_1
01	X_3
11	X_2

Bitflip Repetition Code: circuit implementation



Syndrome	Correction
00	I
10	X_1
01	X_3
11	X_2

Bitflip Repetition Code: circuit implementation



Syndrome	Correction
00	I
10	X_1
01	X_3
11	X_2

Outline

Stabilizer Formalism: The Pauli Group

- Stabilizer codes are a class of quantum error correcting codes defined by commuting sets of Pauli operators, called the **stabilizer generators**
- Define the Pauli group
  $G_1 = \{\pm I, \pm iI, \pm X, \pm iX, \pm Y, \pm iY, \pm, Z, \pm iZ\} = \langle X, Y, Z \rangle$
- It is enough to consider X, Z together with the prefactors $\pm i$ because $Y = iXZ$
- Any qubit unitary can be written as a linear combination of elements of G
- We also define G_n as n-fold tensor products of elements in G_1
- Notation: $Z_1 Z_3 \equiv Z \otimes I \otimes Z$

The Stabilizer Group - definition [1]

- Consider some subgroup $S \subset G_n$, where all elements **commute**
- Let V_S be a $2k$-dimensional subspace of n-qubit states defined by $s|\psi\rangle = +1|\psi\rangle\, \forall s \in S, \forall |\psi\rangle \in V_s$
- This defines a $[[n,k]]$ stabilizer code, which encodes k logical qubits into n physical qubits
- We say S is the **stabilizer group** of V_S, and conversely call V_S the **codespace** stabilized by S
- Example: 3-qubit repetition code [[3, 1]]:
  - Stabilizer group generators: $S = \langle Z_1 Z_2, Z_2 Z_3 \rangle$
  - Codespace: $V_S = \{\alpha|000\rangle + \beta|111\rangle \,|\alpha,\beta \in \mathbb{C}, |\alpha|^2 + |\beta|^2 = 1\}$

[1] D. Gottesmann, Stabilizer codes and quantum error correction, arXiv:quant-ph/9705052

Stabilizer Codes: Error Detection and Correction

- Say some error $g \in G_n$ occurs on $|\psi\rangle \in V_S$. Since elements of G_n either **commute** or **anti-commute** with each other, g will either commute or anti-commute with each stabilizer in S
- If it **anti-commutes** with at least one stabilizer, it is a **detectable error**
- If it **commutes** with all stabilizers and is not itself a stabilizer, it is a **non-detectable** error (logical operator)
- Example: 3-qubit repetition code:
  - $\{X_1, Z_1 Z_2\} = 0$ so X_1 is a detectable error
  - $[X_1 X_2 X_3, Z_1 Z_2] = [X_1 X_2 X_3, Z_2 Z_3] = 0$ so $X_1 X_2 X_3$ is a non-detectable (i.e. logical error). In fact it is logical X in this code
- Measuring all the stabilizer generators on logical state $|\psi\rangle$ will give us a **syndrome** that we then use to apply the corresponding correction (analogous to classical parity checks) (see 3-qubit repetition code circuit from earlier)

Calderbank-Shor-Steane (CSS) codes

- In general, stabilizer generators can have mixed elements e.g. $X_1 Z_2 Z_3 X_4 \ldots$
- CSS codes are a "nice" type of stabilizer codes built by taking the parity check matrices H_X and H_Z of 2 classical codes C_1 and C_2 to define the X and Z stabilizers respectively. The generators are thus only pure X or pure Z operators (see hands-on session)
- Syndrome measurement: to measure a qubit in the X basis we need to apply a Hadamard transform to the qubit since $\langle\psi| HZH |\psi\rangle = \langle\psi| X |\psi\rangle$

Outline

Transversal gates in CSS codes

- Transversal gates, gates that can be written as a tensor product of gates inside each **code block**, are a type of fault-tolerant gates
- All Clifford gates are transversal in CSS codes
- For example in some CSS codes the *CNOT* gate on logical qubits 1 and 2 can be implemented by applying a *CNOT* gate between each homologous qubit of code blocks 1 and 2

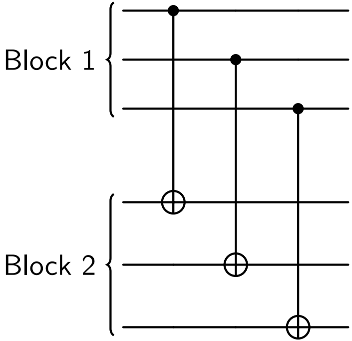


Figure: Transversal CNOT gate implementation for a 3-qubit CSS code

Transversal gates and error spread

An operation is said to be **fault-tolerant** if it does not increase the weight of an error $w(e)$ = the number of qubits that e affects within one **code block**.

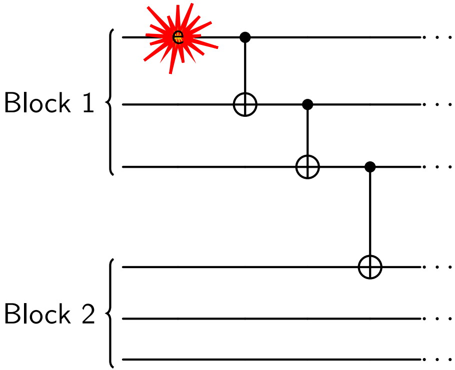


Figure: Non-fault tolerant CNOT gate implementation for a 3-qubit code

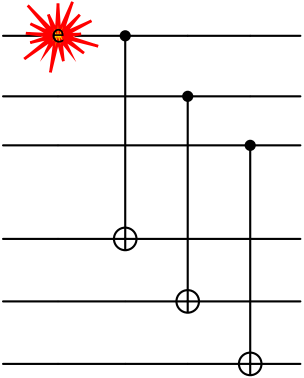


Figure: Transversal CNOT gate

Eastin-Knill Theorem

- **Theorem**: There is no non-trivial local-error-detecting quantum error correcting code that admits a universal set of transversal gates[2]. :(
- But transversal is not the only fault-tolerant construction!

[2] Eastin, B., Knill, E. (2009). Restrictions on transversal encoded quantum gate sets. Physical review letters, 102(11), 110502.

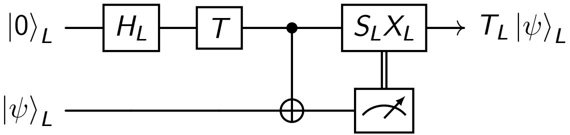
Universal Quantum Computing with Logical Qubits

Knill-Gottesman: Clifford-circuits efficiently simulable

- Generated by $\{H, S, \mathrm{CNOT}\}$ gates
- Many codes allow transversal implementation

Non-Clifford (e.g. T-gate), required for universal gate set.

- **Eastin-Knill**: No transversal implementation for CSS codes
- Requires magic state preparation and teleportation



Fault-tolerant T-gate

Goal: Apply logical T-gate to state $|\psi\rangle_L = a|0\rangle_L + b|1\rangle_L$
Need ancilla qubit. T gate is applied transversally $\rightarrow$ Does not correspond to logical T-state.

State before measurement

$$\frac{1}{\sqrt{2}}(a|0\rangle + be^{i\pi/4}|1\rangle)|0\rangle + (b|0\rangle + ae^{i\pi/4}|1\rangle)|1\rangle$$

If we measure $|0\rangle$, we are done, otherwise apply correction SX.
Preparation of ancilla has to be done fault-tolerantly!

Threshold Theorem

- Reliable quantum computation is possible if the physical error rate p is below a certain **threshold** p_{th}.
- For $p < p_{\text{th}}$, error is **exponentially suppressed** as we scale the code.

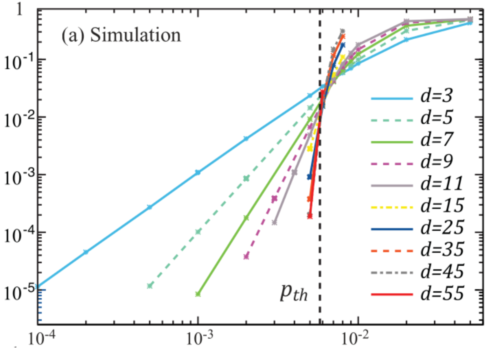


Figure: Exponential suppression as we scale the code

Outline

Surface Code - Introduction

- 2D stabilizer code proposed by Kitaev et al. [7]
  Belongs to the class of CSS codes
- Pauli-Z and Pauli-X type checks
- Planar graph connectivity
  Ideal for superconducting circuits
- High threshold ($\sim$1%) against noise
- Parallel syndrome extraction

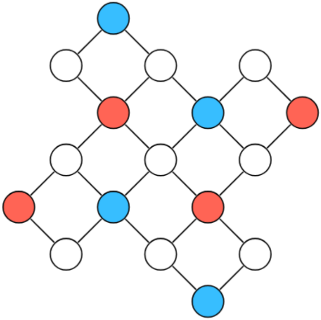


Figure: Surface code with 9 data and 8 ancilla qubits [15].

Surface Code - Stabilizers

Stabilizers at the interior of the surface check 4 qubits at a time. For each group (called *plaquette*) we have:

$$S_X^{(i)} = X_i X_{i+1} X_{i+2} X_{i+3} \qquad S_Z^{(i)} = Z_i Z_{i+1} Z_{i+2} Z_{i+3}$$

Detect an odd number of X/Z errors per plaquette.

Order of CNOT gates matters to avoid *hook errors*.

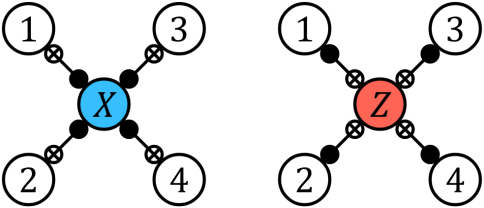


Figure: Pauli-X and Pauli-Z type stabilizers for the surface code. CNOT gate schedule measuring syndrome indicated by vertex index [15].

Surface Code - Error Classification

Only need to correct Pauli errors: $E = P_1 \otimes \ldots \otimes P_n$ where $P_i \in \{I, X, Y, Z\}$

Detectable Errors

- Anti-commute with stabilizers: $\exists S \in \mathcal{S} : SE = -ES$
- Example: Single-qubit errors

Undetectable errors

- Product of stabilizers: $E = S_1 \ldots S_n,\ S_i \in \mathcal{S}$
- Logical operators: Normalizers of $\mathcal{S}$

Beauty of surface code: Errors have *topological* interpretation!

Surface Code - Detectable Errors

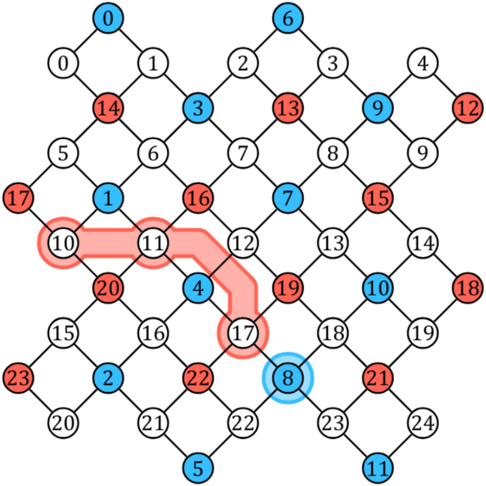


Figure: Detectable chain of Pauli-Z errors [15].

Surface Code - Detectable Errors

Error Chain Properties:

- Errors manifest as **chains** on surface
- Chain endpoints flagged by syndromes:
  - One syndrome if chain ends at **boundary**
  - Two syndromes for **interior** chains
- Pauli-Y triggers 4 syndromes Equivalent to X and Z errors

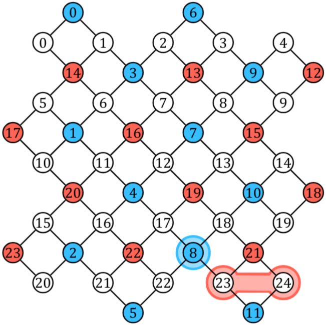


Figure: Detectable chain of Pauli-Z errors [15].

Surface Code - Undetectable Errors

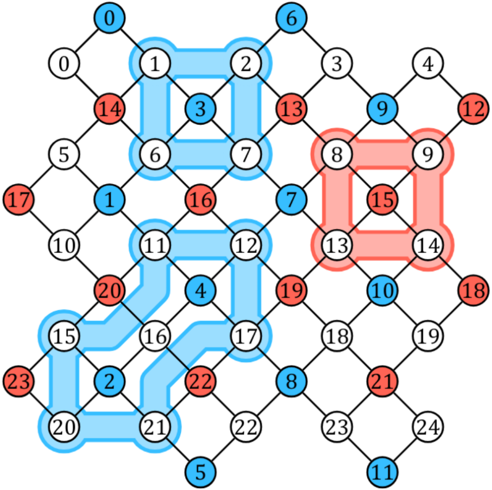


Figure: Undetectable errors which are products of stabilizers generators [15].

Surface Code - Logical Gates

Logical Gates Properties:

- Connect opposite borders
- Unique up to stabilizer product
- Anti-commuting logical operators cross odd number of times

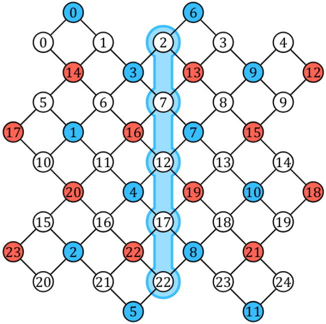


Figure: Chain of Pauli-X forming logical X_L operator [15].

Surface Code - Logical Gates

Logical Gates Properties:

- Connect opposite borders
- Unique up to stabilizer product
- Anti-commuting logical operators cross odd number of times

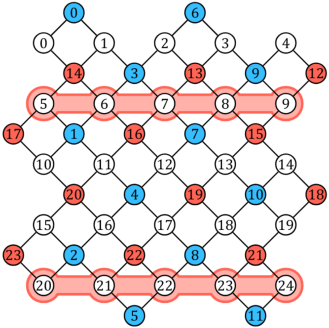


Figure: Chains of Pauli-Z forming logical Z_L operator [15].

Surface Code - Logical Gates

Logical Gates Properties:

- Connect opposite borders
- Unique up to stabilizer product
- Anti-commuting logical operators cross odd number of times

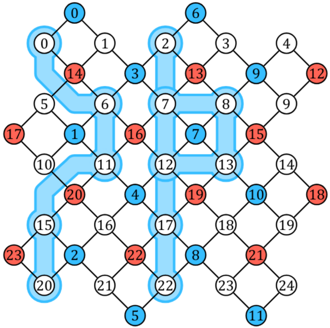


Figure: Equivalent logical X_L chains [15].

Surface Code - Code Distance

Question: How many errors can we correct?

For d^2 data qubits, shortest logical error chain has length d.
$\rightarrow$ We can correct up to $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors.

The surface code is a $[[d^2, 1, d]]$ CSS code with code distance d.

Surface Code - Entangling Gates

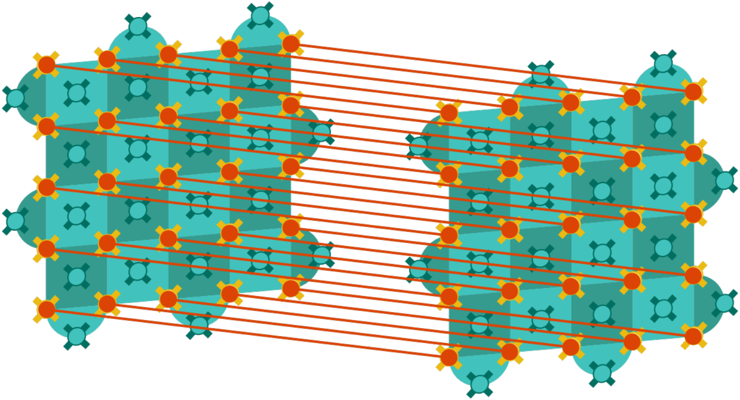


Figure: **Transversal** logical CNOT with pairwise matching physical qubits. Suitable for neutral atom or ion-trapped architectures where qubits can be moved [4].

Surface Code - Entangling Gates

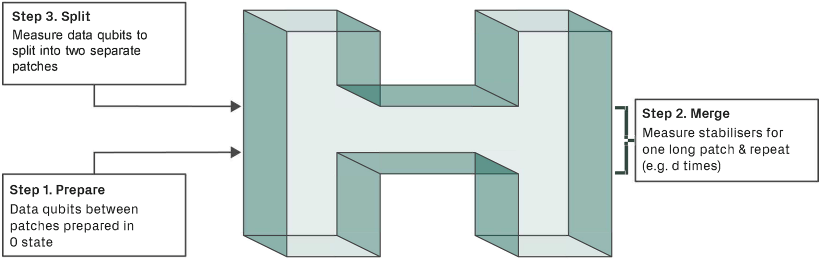
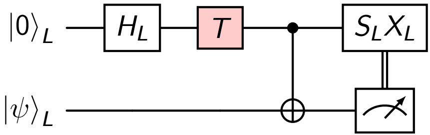


Figure: Logical CNOT through **lattice surgery**. Involves an ancilla surface code patch and stabilizer measurement along adjacent surface edges [4].

Surface Code - T-gate (via state-injection)

T-gate prepared via state teleportation.

Need **magic state**: $T_L \left|+\right\rangle_L$



Protocol for faulty state-injection [12]:

- Prepare physical qubit in state $\left|\psi\right\rangle$
- Initialize small distance surface code (e.g. $\hat{d} = 3$) in $\left|0\right\rangle_L$
- Spread state via CNOT operations
- Protect state with syndrome measurement rounds
- Grow to target distance: $\hat{d} \to d$

Not fault-tolerant: Low-distance $\hat{d}$ allows for errors

Surface Code - State injection example

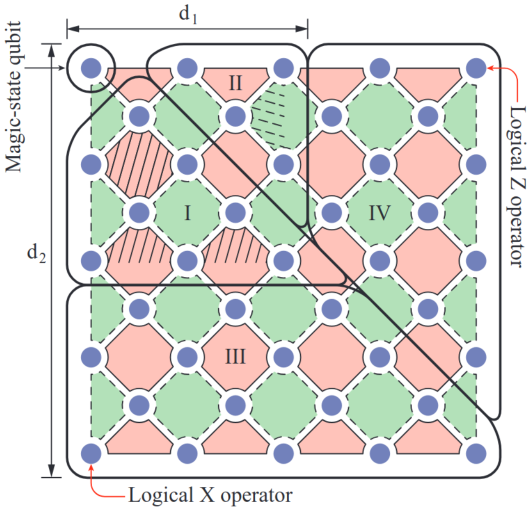


Figure: Preparation of magic state by preparing a single physical qubit and growing the surface code distance [12].

Surface Code - Magic State Distillation

Muller-Reed $[[15, 1, 3]]$**-code**: Smallest code with transversal T-gate

Example: 15-to-1 protocol

1. Encode by measuring stabilizers
2. Apply transversal (faulty) T-gate
   Surface code: Via state-injection
3. Measure Stabilizers
   Detect up to weight-3 errors
4. Discard and repeat, if errors detected

Output: Magic state $T_L \left|+\right\rangle_L$

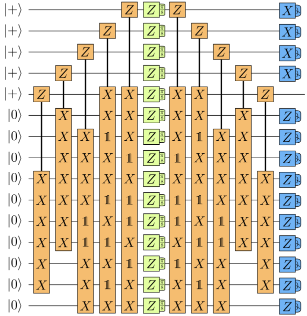


Figure: 15-to-1 magic state distillation protocol [13].

If error probability of T-gate is p_{in}, success probability is $p_{out} = 35p_{in}^3$.
Further distillation rounds can use teleportation for transversal T_L.

Fault-Tolerant Quantum Architecture

Based on surface code with Clifford+T gate set. Gates are implemented using fault-tolerant lattice surgery.

Core Processor Components

1. **Memory Fabric**
   - Data storage
   - Performs logical operations
2. **Magic State Buffer**
   - Stores prepared T-states
   - Enables on-demand T-gates
3. **Magic State Factory**
   - 15:1 distillation protocol
   - Continuous state preparation

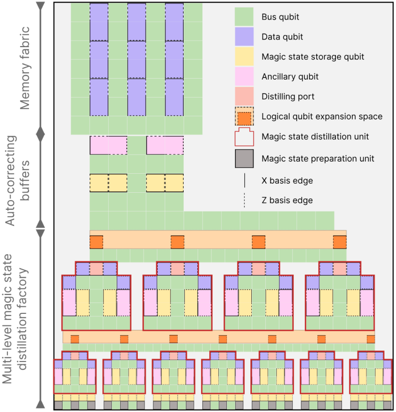


Figure: FTQC architecture [16].

Fault-Tolerant Quantum Computing - T-count

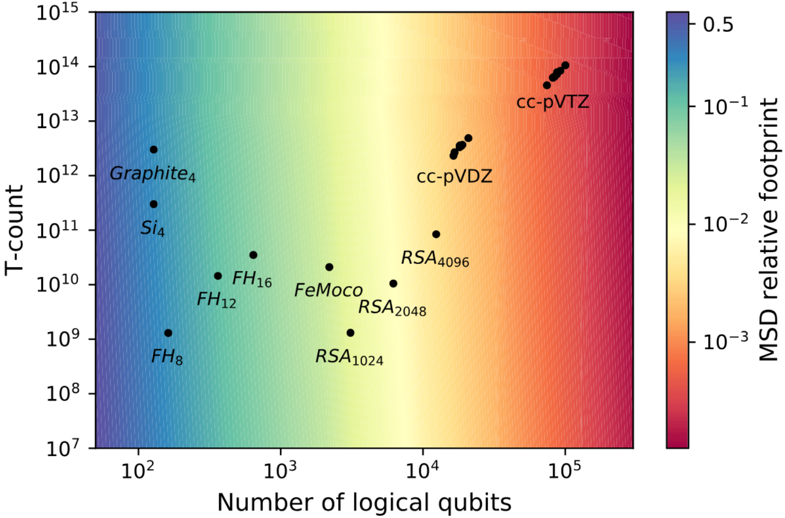


Figure: Ratio of magic state distillation (MSD) footprint to total computational footprint for different number of logical qubits and T-counts for fusion-based QC [11].

Fault-Tolerant Quantum Computing - Resource Estimation

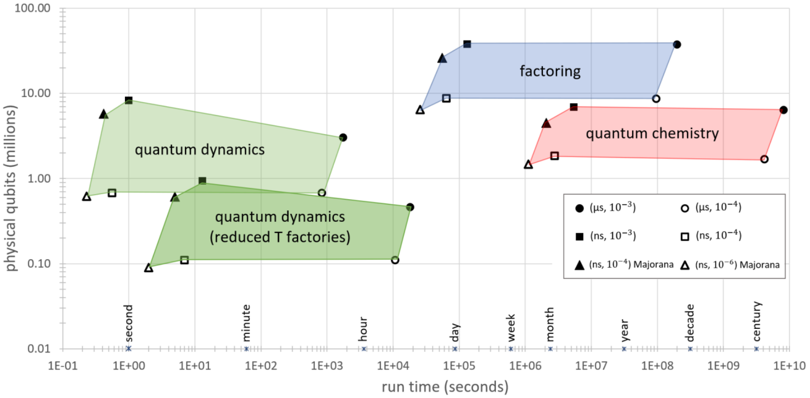


Figure: Runtime of 3 applications for different gate times and modalities: superconducting [ns], ion-traps [μs], and Majorana [3].

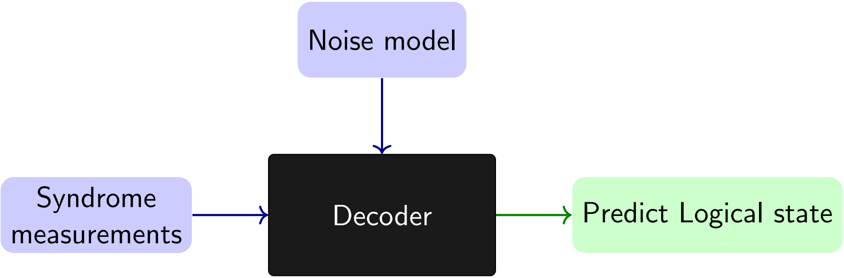
Observation: With 100 [μs] gate times, large algorithms will take almost a year!

Outline

The Decoder

Goal: Determine the state of the logical qubit

Input: Syndrome measurements and noise information
Output: Logical state estimate



The Backlog Problem

Non-Clifford operations (e.g. T-gates) require processing of *all* prior syndrome measurements

When Decoder slower than syndrome generation rate:

1. Define rates:
   - r_{gen}: syndrome generation rate
   - r_{proc}: syndrome processing rate
2. Let $f = \frac{r_{gen}}{r_{proc}} \geq 1$ (backlog factor)
3. For initial T-gate (T_0):
   - Processing overhead time: Δ_{gen}
   - New syndromes during processing: $D_1 = r_{gen} \times \Delta_{gen}$

Observation

Terhal [17] showed: Overhead for k-th T-gate grows as $f^k D_1$

The Backlog Problem

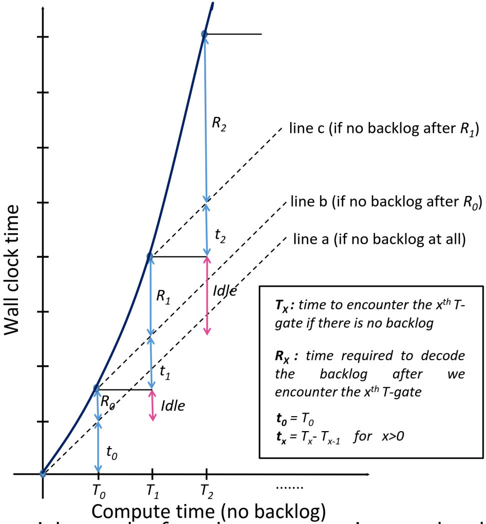


Figure: Exponential growth of syndrome processing overhead for $f > 1$ [10].

The Backlog Problem - Example

Circuit Parameters

- Logical qubits: 100
- Total gates: 2,356
- T-gates: 686

Timing Parameters

- Syndrome generation cycle: 400 [ns]
- Decoder processing time: 800 [ns]
- Backlog factor $f = \frac{r_{gen}}{r_{proc}} = 2$

Circuit execution time: $\mathbf{10^{196}}$ seconds!

Decoder speed and **T-gate count** critical metrics for practical quantum computation.

Real-Time Decoding for Superconducting QPU

Real-time decoding challenging for superconducting devices due to gate speed: Cycle time < 1 [μs].

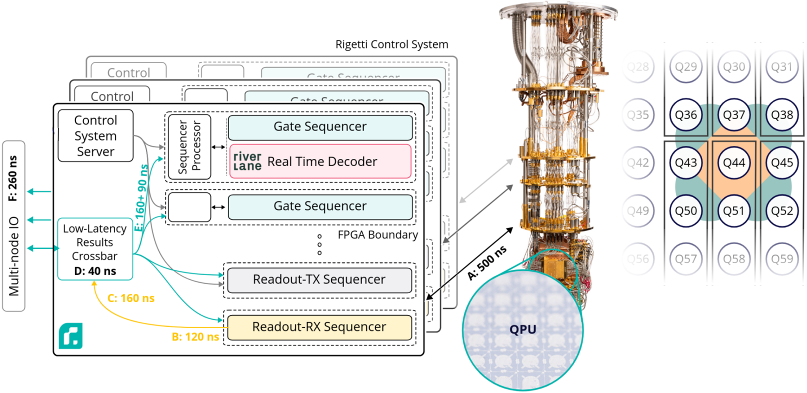


Figure: Integration of Riverlane's FPGA decoder into Rigetti's control system. Latencies are represented by edge labels. Demonstrate mean decoding time below 1 [μs] for Rigetti's Ankaa-2 device [5].

Type of Decoders

Many types of decoders exist, with unique properties:

- Maximum-likelihood decoder
  Optimal, but computationally infeasible
- Matching-based: (e.g. MWPM [9] or BP)
  Optimal for independent errors, widely studied
- Clustering-based (e.g. Union Find [6])
  Fast, near-linear time complexity
- Tensor Networks:
  Handles correlations well, higher computational overhead
- Neural Networks: (e.g. AlphaQubit [2])
  Potential for handling complex noise models

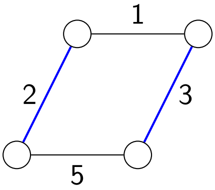
Key trade-off: **decoding speed** vs. **correction accuracy**

We are going to explore *MWPM* and *neural decoders*.

Graph Matching

Perfect Matching Problem: Given a weighted graph $G = (V, E, w)$, where $w : E \to \mathbb{R}$

- Find matching $M \subseteq E$ where each $v \in V$ appears in exactly one edge in M
- Minimize total weight: $\min_M \sum_{e \in M} w(e)$



MWPM Decoder - Idea

Observation: Error chains create distinct syndrome patterns

Types of Error Chains:

1. *Boundary chains*
   - Single syndrome at interior of surface
   - Other end terminates at code boundary
2. *Interior chains*
   - Two syndromes: one at each end

Matching idea:

- Each chain has an associated occurrence probability
- Match all active syndromes minimizing error probability

MWPM Decoder - Example

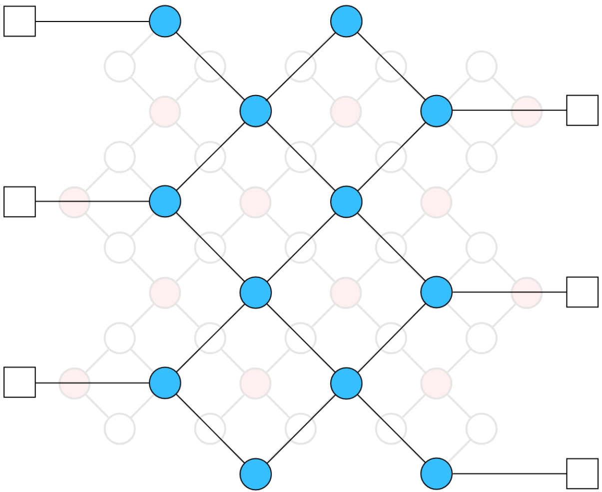


Figure: Tanner graph for Pauli-Z type errors for the distance 5 surface code.

MWPM Decoder - Example

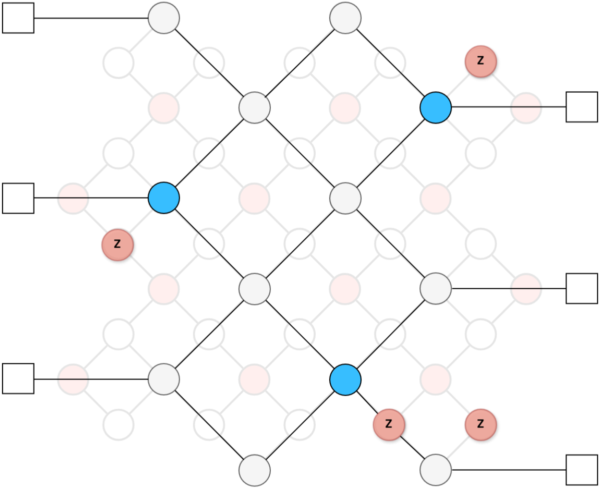


Figure: Active syndromes in Tanner graph for given Pauli-Z errors.

MWPM - Example

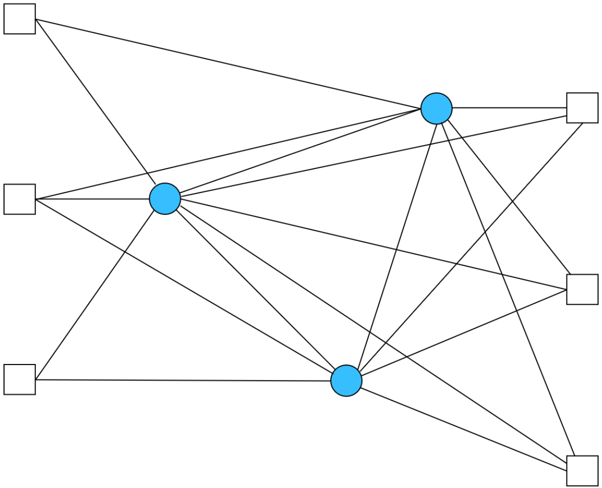


Figure: Syndrome graph for active syndromes.

MWPM - Example

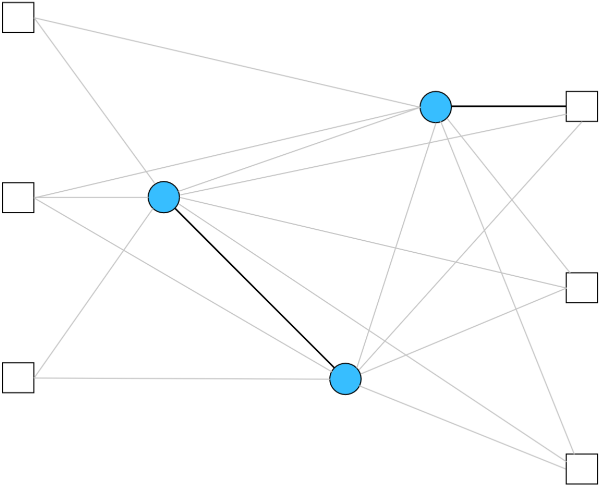


Figure: Matched syndrome graph for active syndromes.

MWPM - Example

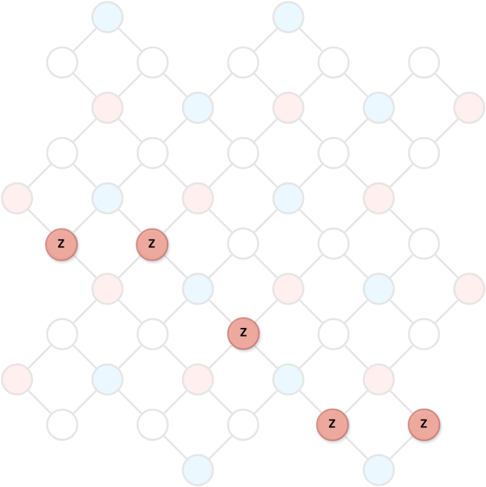


Figure: Decoded errors leading to logical Z_L error by connecting chain of Pauli-Z errors to opposite boundaries.

MWPM Decoder - Construction

Setup:

- Decode Pauli-Z and X errors separately
- Consider **independent** Z errors $E \in \{I, Z\}^n$ for CSS code

For stabilizer generator set $\{S_i\}_i$ define:

- Syndrome bits: $s_i \in \{0, 1\}$, where $s_i = 1$ if generator S_i anti-commutes with E
- Error vector: $e \in \{0, 1\}^n$ if $E_i = Z_i$

Error Probability:

$$p(E) = \prod_i (1 - p_i)^{(1-e_i)} \cdot p_i^{e_i} = \prod_i (1 - p_i) \prod_i \left(\frac{p_i}{1 - p_i}\right)^{e_i}$$

Use logarithmic form, avoiding numerical issues:

$$\log(p(E)) = \sum_i \log(1 - p_i) - \sum_i w_i \cdot e_i,$$

where $w_i = \log((1 - p_i)/p_i)$

MWPM Decoder - Construction

Graph Construction:

- Condition: Each Z-error anti-commutes with two X-stabilizers
- Define matching graph $G = (V, E)$ with $|V| = |s|$
- $(v, w) \in E$, if S_v and S_w anti-commute with Pauli-Z on qubit
- Set edge weight to w_i for qubit i

Decoding Strategy:

- *Perfect matching*: Match all nodes with $s_i = 1$
- *Minimum-weight*: Find smallest chain with $s_i = 1$ at boundaries
  $\rightarrow$ More probable errors have lower weight

Implementation:

- Matching: Edmond's Blossom algorithm
  - Complexity: $\mathcal{O}(|s|^3 \log(|s|))$
- Syndrome graph: Dijkstra's algorithm

Neural Network Decoder - AlphaQubit

QEC's "The Bitter Lesson" moment?

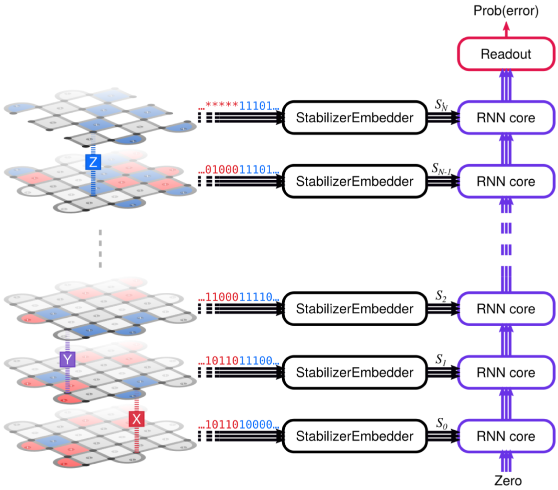


Figure: Decoder's recurrent network structure. Syndromes update transformer state. Outputs single-bit, indicating if logical bit was flipped. Evaluated up to code distance 11 [2].

AlphaQubit - Training

Pretraining

- 2.5 billion samples from 3 sources:
  1. SI1000: 25 QEC rounds, no device-fit
  2. Noise estimate from XEB
  3. Noise estimate for Tanner graph weights p_{ij}

Finetuning

- Pauli+ simulator including leakage, analogue readouts, and cross-talk
- 100 million samples

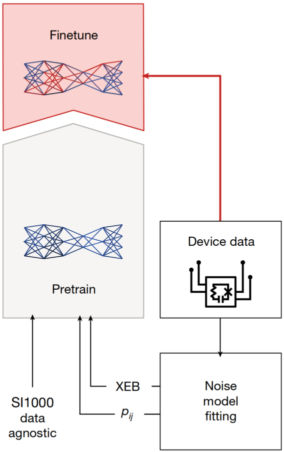


Figure: Training stages [2].

AlphaQubit - Stabilizer Embedding Layer

Input:

- Binary syndrome measurement and temporal differences
- Leakage events and their probability
- Embedded stabilizer index i

Output: $d^2 - 1$ different embeddings

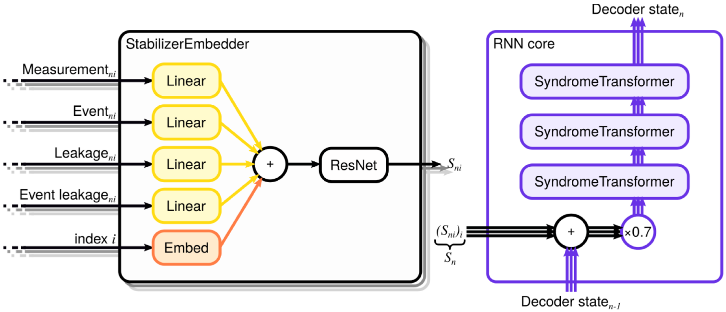


Figure: Stabilizer embeddings used as input for AlphaQubits internal transformer state update round [2].

AlphaQubit - Results

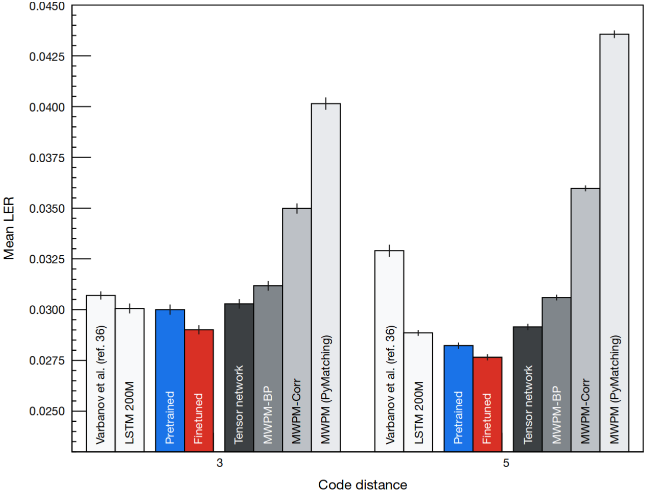


Figure: Mean logical error per QEC round for Surface Code distances 3 and 5 on Google's Sycamore device. Results averaged across bases $\{X, Y, Z\}$ [2].

Decoder Threshold Analysis

Threshold Dependencies:

- Decoder algorithm
- Noise model
- QEC code structure

For distance d, physical error p and threshold p_{thr}:

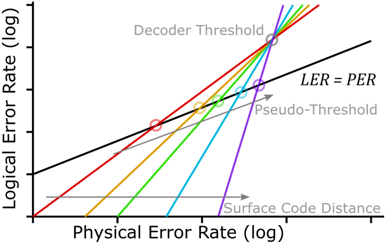


Figure: Threshold example [15].

Logical Error Scaling:

$$\varepsilon_d \propto \left(\frac{p}{p_{thr}}\right)^{\frac{(d+1)}{2}}$$

Error Suppression:

$$\Lambda = \frac{\varepsilon_d}{\varepsilon_{d+2}} \sim \frac{p_{thr}}{p}$$

Note: Threshold comparisons must consider all factors!

Outline

Google - Surface Code Experiment

Quantum Memory Experiment:
Preserve logical qubit for many QEC cycles

Setup:

- Sycamore: 105-qubits transmon device
- Distances: $d = 3$, 5, and 7
- X/Y 25 [ns], CZ 42 [ns]
- TLS mitigation strategy

Main Results:

- Demonstrate $\Lambda > 2$
- Life-time of logical qubit $2\times$ of best physical qubit on QPU
- Real-time decoding $< 1.1[\mu s]$

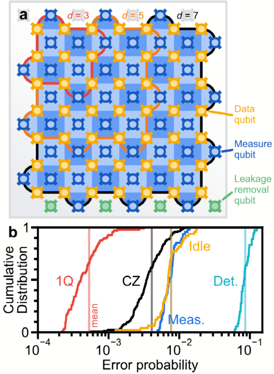


Figure: **a** Sycamore topology. **b** Gate error distribution [1].

Google - Real-Time Decoder Data Flow

1. Control electronics classify I/Q readout into 0/1
2. Transmitted to workstation via low-latency Ethernet
3. Measurements converted to detection events
4. Streamed to constant sized shared-buffer
5. Decoder reads from buffer

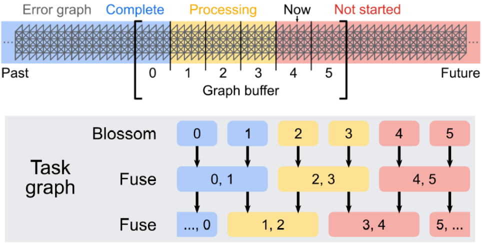


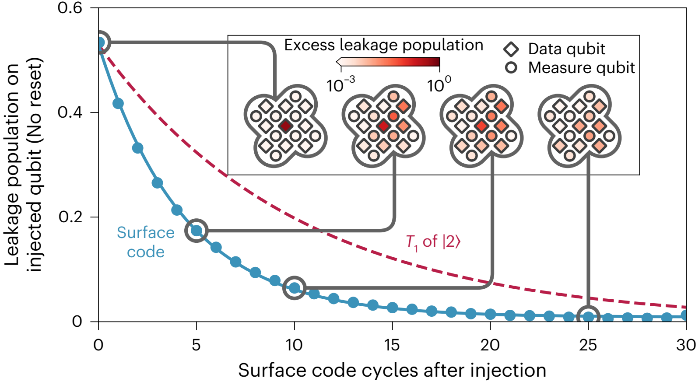
Figure: Windowed streaming decoder: Local Blossom algorithm with subsequent fusing until global MWPM is found [1].

Google - Correlated Errors through Leakage

Transmons not ideal qubits → Leakage to $|2\rangle, |3\rangle, \ldots$ possible.

Problem: QEC assumes uncorrelated errors. Leakage causes correlated errors! Especially **CZ gate** prone to leakage.

DQLR: Use *Leakage iSWAP* to transfer leakage to ancillas [14].



Google - Results

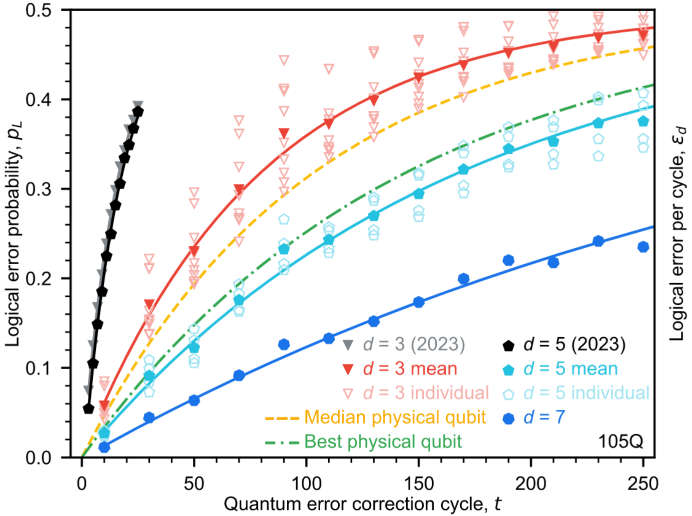


Figure: Logical error rates (LER) over multiple QEC cycles demonstrating $\Lambda = 2.14 \pm 0.02$ [1].

Quantum Error Correction - Summary

Many more topics ...

- Quantum LDPC codes, color codes, ...
- Subsystem codes
- Bosonic codes
- Quantum resource estimation
- ...

An Interdisciplinary Field!

- QPU fabrication and control
- Software development and tooling
- Novel error correction code design

Theory Meets Practice

- Transition from theory to implementation
- Emerging real-world demonstrations

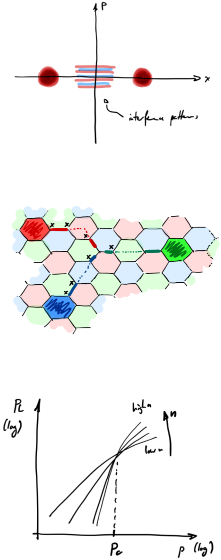


Figure: Source: [8]

References I

R. Acharya, L. Aghababaie-Beni, I. Aleiner, T. I. Andersen, M. Ansmann, F. Arute, K. Arya, A. Asfaw, N. Astrakhantsev, J. Atalaya, R. Babbush, D. Bacon, B. Ballard, J. C. Bardin, J. Bausch, and et al.
Quantum error correction below the surface code threshold, 2024.

J. Bausch, A. W. Senior, F. J. H. Heras, M. Newman, C. Gidney, C. Jones, H. Neven, S. Blackwell, and D. Kafri.
Learning high-accuracy error decoding for quantum processors.

Nature, 2024.

M. E. Beverland, P. Murali, M. Troyer, K. M. Svore, T. Hoefler, V. Kliuchnikov, G. H. Low, M. Soeken, A. Sundaram, and A. Vaschillo.
Assessing requirements to scale to practical quantum advantage, 2022.

References II

E. Campbell.
How to develop the quantum error correction stack for every qubit, 2024.

L. Caune, L. Skoric, N. S. Blunt, A. Ruban, J. McDaniel, J. A. Valery, A. D. Patterson, A. V. Gramolin, J. Majaniemi, K. M. Barnes, T. Bialas, O. Buğdaycı, O. Crawford, G. P. Gehér, H. Krovi, E. Matekole, C. Topal, S. Poletto, M. Bryant, K. Snyder, N. I. Gillespie, G. Jones, K. Johar, E. T. Campbell, and A. D. Hill.
Demonstrating real-time and low-latency quantum error correction with superconducting qubits, 2024.

N. Delfosse and N. H. Nickerson.
Almost-linear time decoding algorithm for topological codes.
Quantum, 5:595, Dec. 2021.

References III

E. Dennis, A. Kitaev, A. Landahl, and J. Preskill.
Topological quantum memory.
Journal of Mathematical Physics, 43(9):4452–4505, Sept. 2002.

P. Faist.
Quantum error correction.
Lecture Notes, aug 2024.

O. Higgott.
Pymatching: A python package for decoding quantum codes with minimum-weight perfect matching, 2021.

References IV

A. Holmes, M. R. Jokar, G. Pasandi, Y. Ding, M. Pedram, and F. T. Chong.
Nisq+: Boosting quantum computing power by approximating quantum error correction.
2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA), pages 556–569, 2020.

I. H. Kim, Y.-H. Liu, S. Pallister, W. Pol, S. Roberts, and E. Lee.
Fault-tolerant resource estimate for quantum chemical simulations: Case study on li-ion battery electrolyte molecules.
Physical Review Research, 4(2), Apr. 2022.

Y. Li.
A magic state's fidelity can be superior to the operations that created it.
New Journal of Physics, 17(2):023037, Feb. 2015.

References V

D. Litinski.
A game of surface codes: Large-scale quantum computing with lattice surgery.
Quantum, 2018.

K. C. Miao, M. McEwen, J. Atalaya, D. Kafri, L. P. Pryadko, A. Bengtsson, A. Opremcak, K. J. Satzinger, Z. Chen, P. V. Klimov, C. Quintana, and e. a. Acharya.
Overcoming leakage in quantum error correction.
Nature Physics, 19(12):1780–1786, Oct. 2023.

R. W. J. Overwater, M. Babaie, and F. Sebastiano.
Neural-network decoders for quantum error correction using surface codes: A space exploration of the hardware cost-performance tradeoffs.
IEEE Transactions on Quantum Engineering, 3:1–19, 2022.

References VI

A. Silva, A. Scherer, Z. Webb, A. Khalid, B. Kulchytskyy, M. Kramer, K. Nguyen, X. Kong, G. A. Dagnew, Y. Wang, H. A. Nguyen, K. Olfert, and P. Ronagh.
Optimizing multi-level magic state factories for fault-tolerant quantum architectures, 2024.

B. M. Terhal.
Quantum error correction for quantum memories.
Reviews of Modern Physics, 87(2):307–346, Apr. 2015.