



Limitations of Fault-Tolerant Quantum Computers in Chemistry

QCPC 23

David Herrera Martí (feat. Zach Blunden-Codd)

CEA List

20/01/23

Summary

PART 1

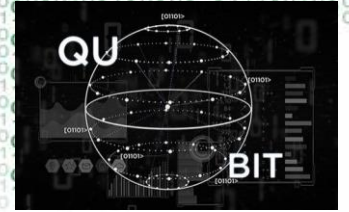
Introduction to Fault-Tolerant Quantum Computing

PART 2

Quantum Phase Estimation Algorithm

PART 3

Limitations of QPE in Quantum Chemistry



Fault-Tolerant Quantum Computing

Limitations of NISQ architectures

Stabiliser codes

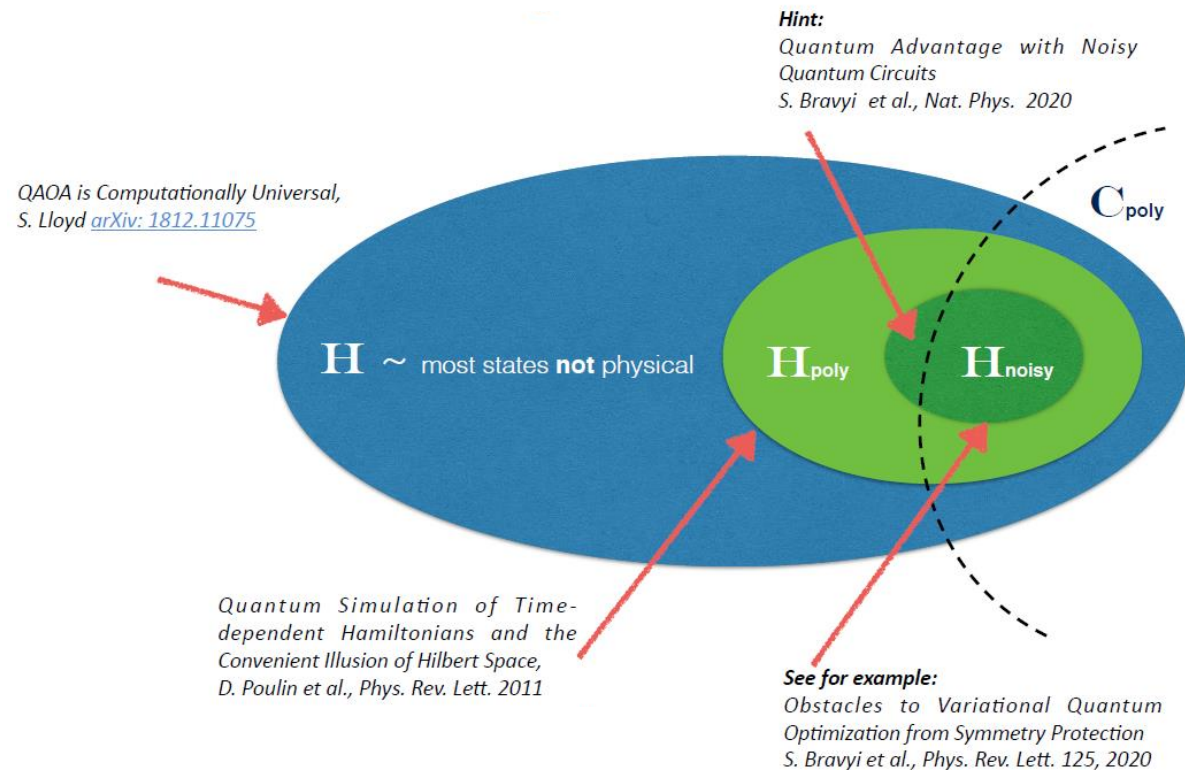
Fault tolerance. Transversality. Threshold Theorem

Surface Code

Limitations of NISQ architectures

Expressivity / Trainability

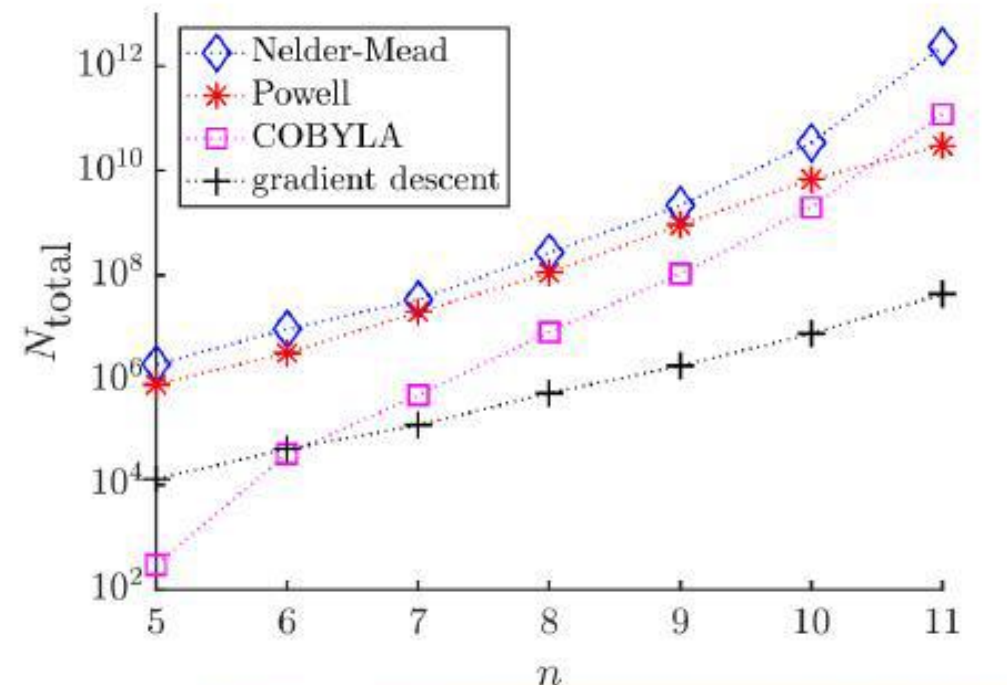
Variational Quantum algorithms have two main drawbacks:



Limited computational power

Noise imposes an upper bound on circuit depth and on maximum algorithmic complexity.

→ Only constant-depth circuits are realistic in NISQ architectures



From [Effect of barren plateaus on gradient-free optimization](#)
Arrasmith et al. Quantum Journal (2021)

Trainability Issues

Resources (# experiment repetitions: N_{total}) for training variational algorithms increase exponentially on the number of qubits (n) for a fixed accuracy.

Fault Tolerance

Error Correction

To go beyond NISQ, extended coherence times are needed. To extend coherence time, active quantum error correction is needed.

Warmup on (classical) error correction.

- The core idea of error correction is to introduce **redundant bits** to fight noise
- Redundant qubits introduce **correlations** which can tell us something about noise

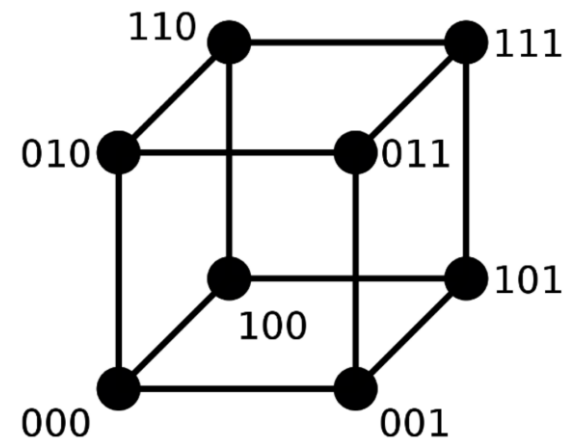
The classical repetition code is the following encoding $0 \rightarrow [000]$ $1 \rightarrow [111]$

The Parity Check matrix \mathbf{C} gives information on whether an error occurred.

$$\mathbf{C} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

For instance $\mathbf{C}[000] = [00]$, $\mathbf{C}[001] = [10]$, $\mathbf{C}[010] = [01]$, $\mathbf{C}[100] = [11]$

But also $\mathbf{C}[111] = [00]$, $\mathbf{C}[110] = [10]$, $\mathbf{C}[101] = [01]$, $\mathbf{C}[011] = [11]$

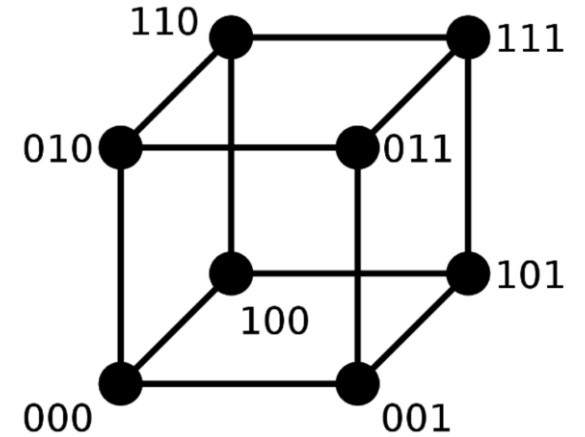


If error probability is low enough (only errors of order less than code distance happen), then correction works.

Fault Tolerance

Bit flip Code (encoding)

Code distance $d = 3$
Only corrects bit flips.



$$|0\rangle_L = |000\rangle \quad |1\rangle_L = |111\rangle$$

$$\frac{\alpha|0\rangle + \beta|1\rangle}{\sqrt{2}} \left\{ \begin{array}{l} \text{---} \bullet \text{---} \bullet \text{---} \\ \text{---} \bigoplus \text{---} \\ \text{---} \bigoplus \text{---} \end{array} \right\} |\psi\rangle = \alpha|000\rangle + \beta|111\rangle = \alpha|0\rangle_L + \beta|1\rangle_L$$

$$X_L = X_1 \otimes X_2 \otimes X_3$$

$$Z_L = \begin{cases} Z_n \\ Z_1 \otimes Z_2 \otimes Z_3 \end{cases}$$

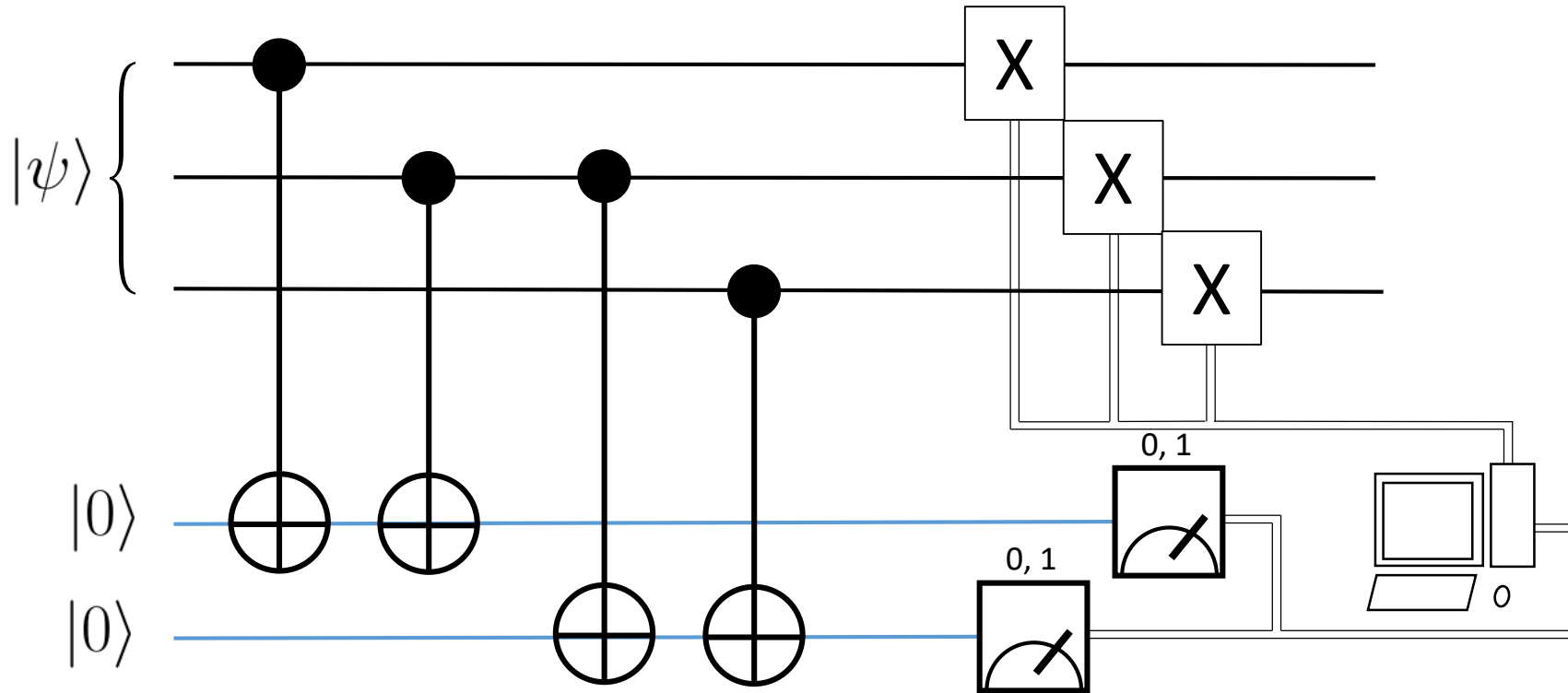
$$X_L |\psi\rangle = \alpha|111\rangle + \beta|000\rangle$$

$$Z_L |\psi\rangle = \alpha|000\rangle - \beta|111\rangle$$

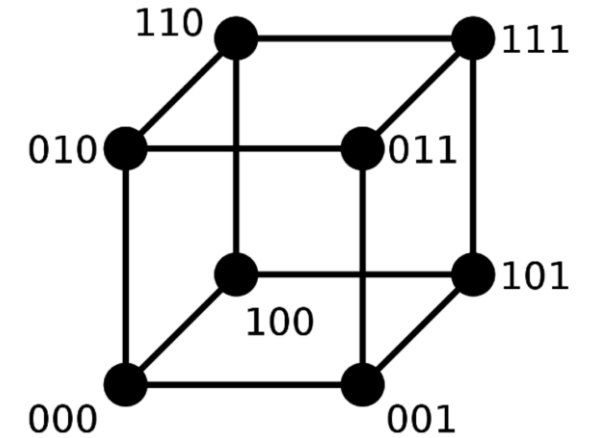
Fault Tolerance

Bit flip Code (correcting)

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



Code distance $d = 3$
Only corrects bit flips.



These three physical qubits make one logical qubit and we will have many such logical qubits in a circuit, which must be corrected independently.

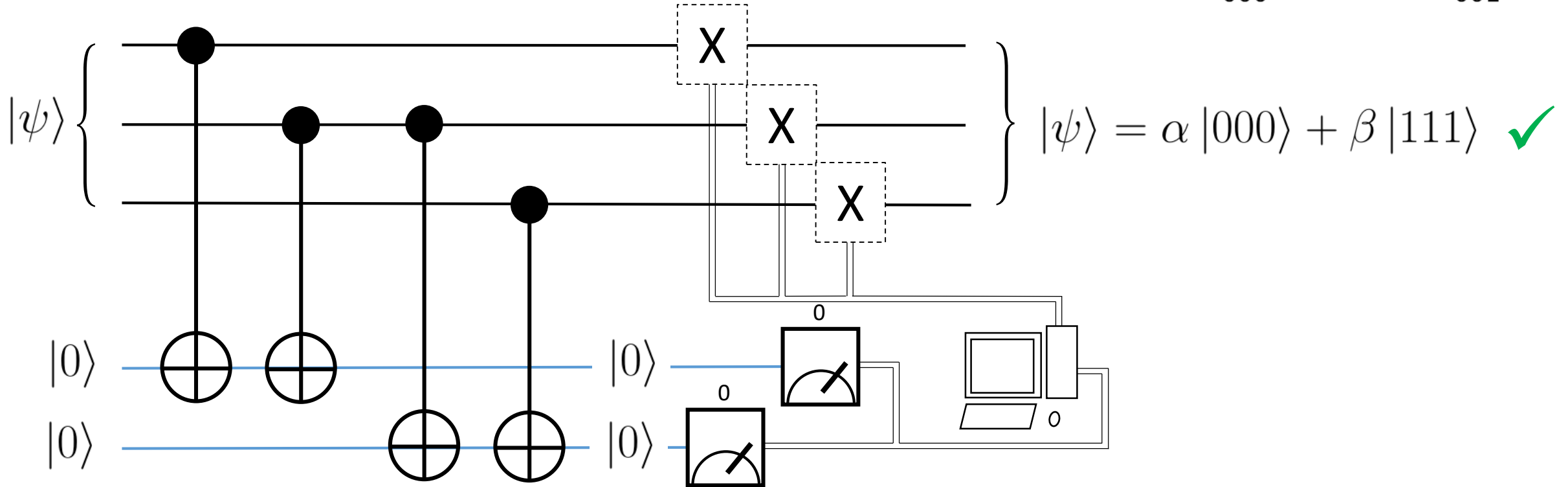
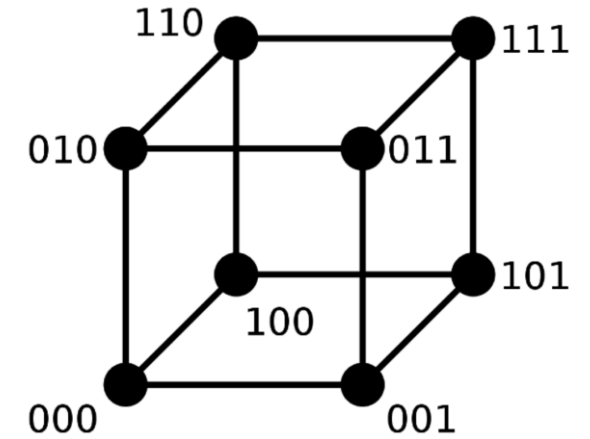
The ancilla qubits used to detect the syndromes (errors) can be reused after measurement.

We can run this error correcting protocol as often as we like on as many of the qubits as we like, but if the gates and measurements involved are imperfect then we may introduce errors into our circuit. The optimal frequency for applying the protocol will depend on the relative sizes of errors in the circuit and the gate and measurement errors in the correction protocol. At present measurement errors are usually dominant.

Fault Tolerance

Bit flip Code (No bit flips)

Code distance $d = 3$
Only corrects bit flips.

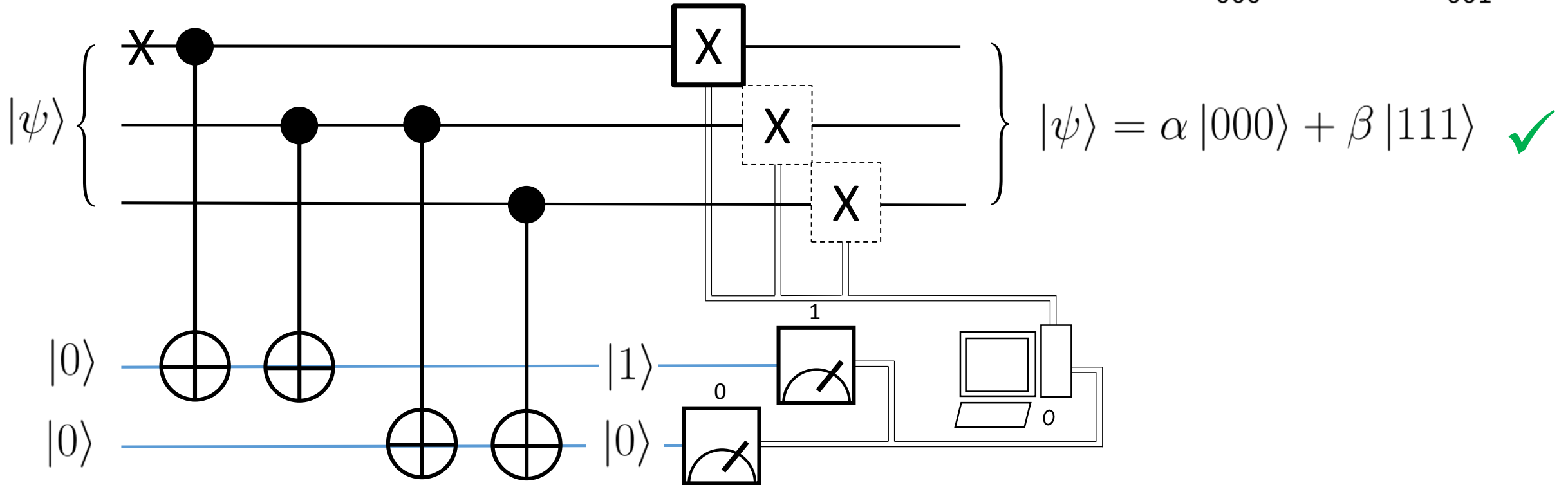
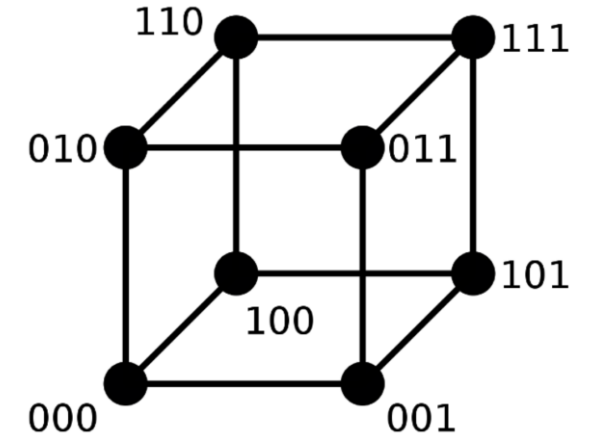


Fault Tolerance

Bit flip Code (One bit flip)

$$|\tilde{\psi}\rangle = \alpha |100\rangle + \beta |011\rangle$$

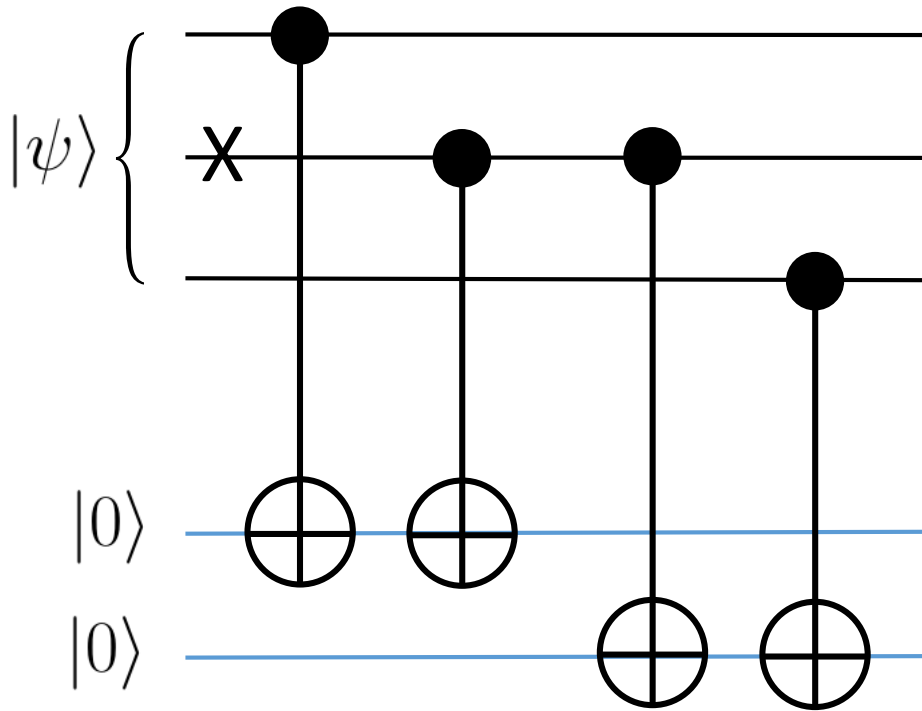
Code distance $d = 3$
Only corrects bit flips.



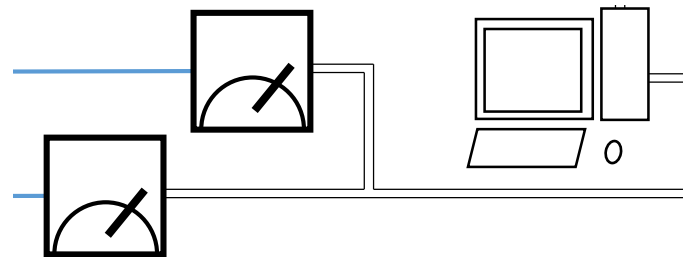
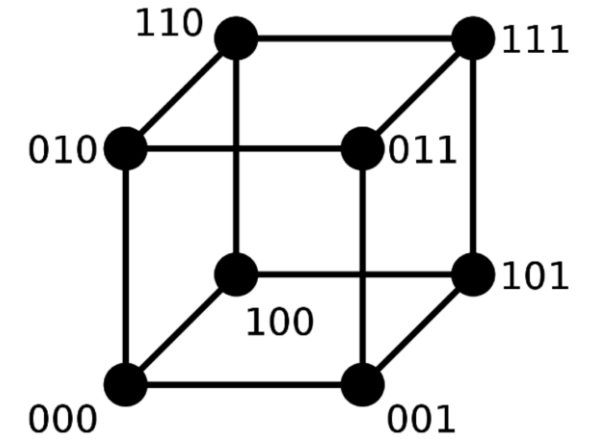
Fault Tolerance

Bit flip Code (One bit flip)

$$|\tilde{\psi}\rangle = \alpha |010\rangle + \beta |101\rangle$$



Code distance $d = 3$
Only corrects bit flips.

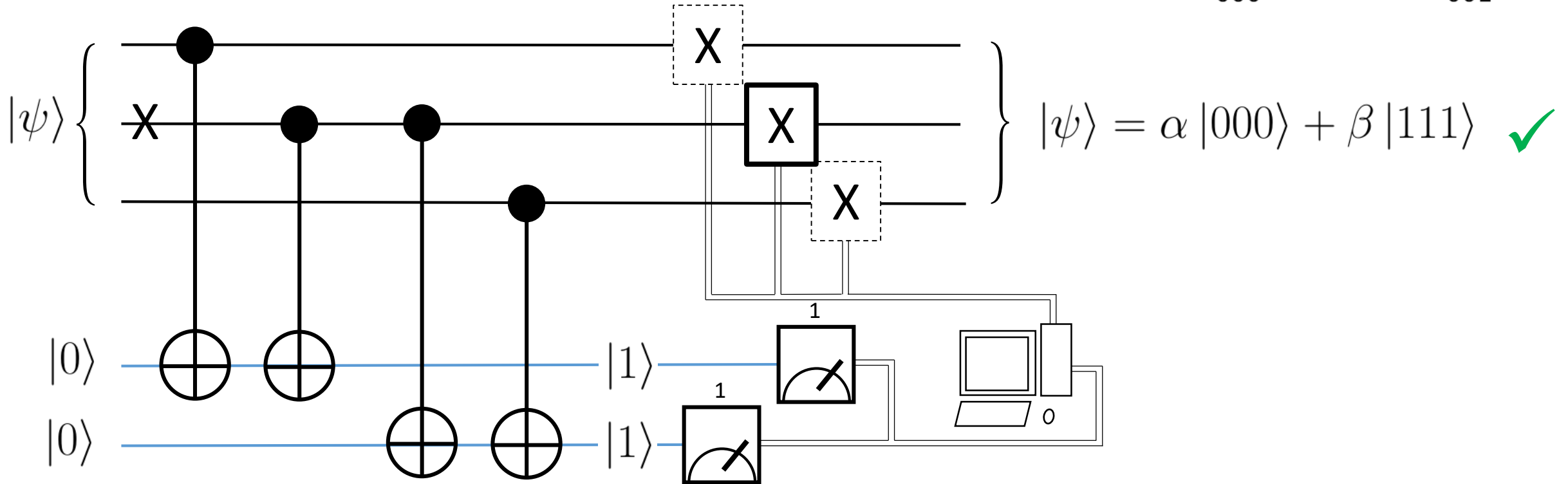
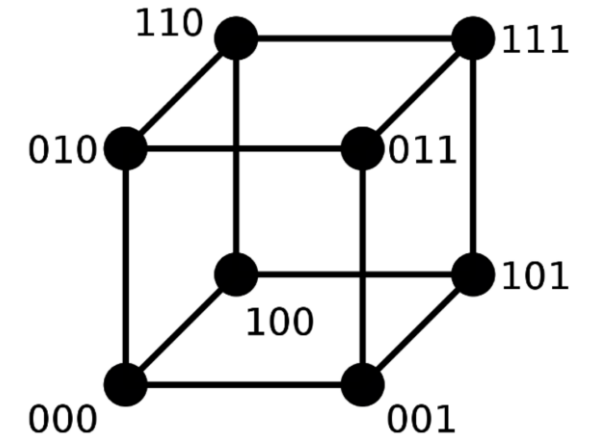


Fault Tolerance

Bit flip Code (One bit flip)

$$|\tilde{\psi}\rangle = \alpha |010\rangle + \beta |101\rangle$$

Code distance $d = 3$
Only corrects bit flips.

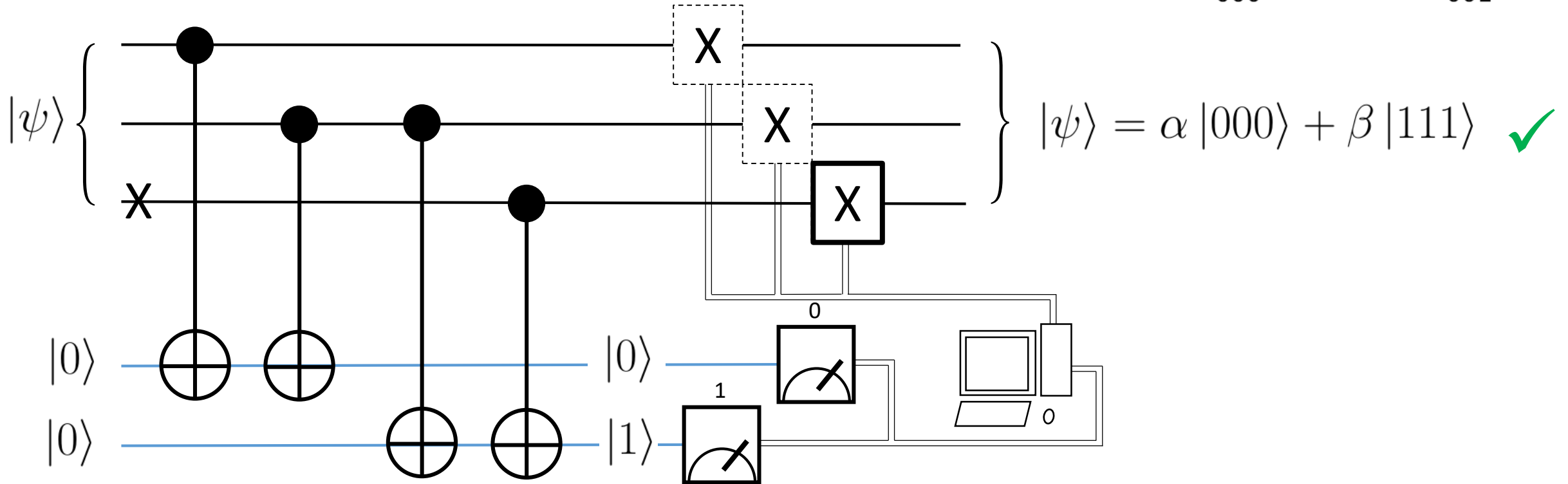
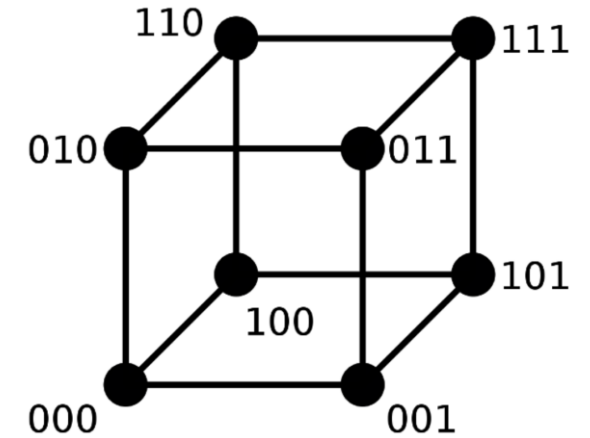


Fault Tolerance

Bit flip Code (One bit flip)

$$|\tilde{\psi}\rangle = \alpha |001\rangle + \beta |110\rangle$$

Code distance $d = 3$
Only corrects bit flips.

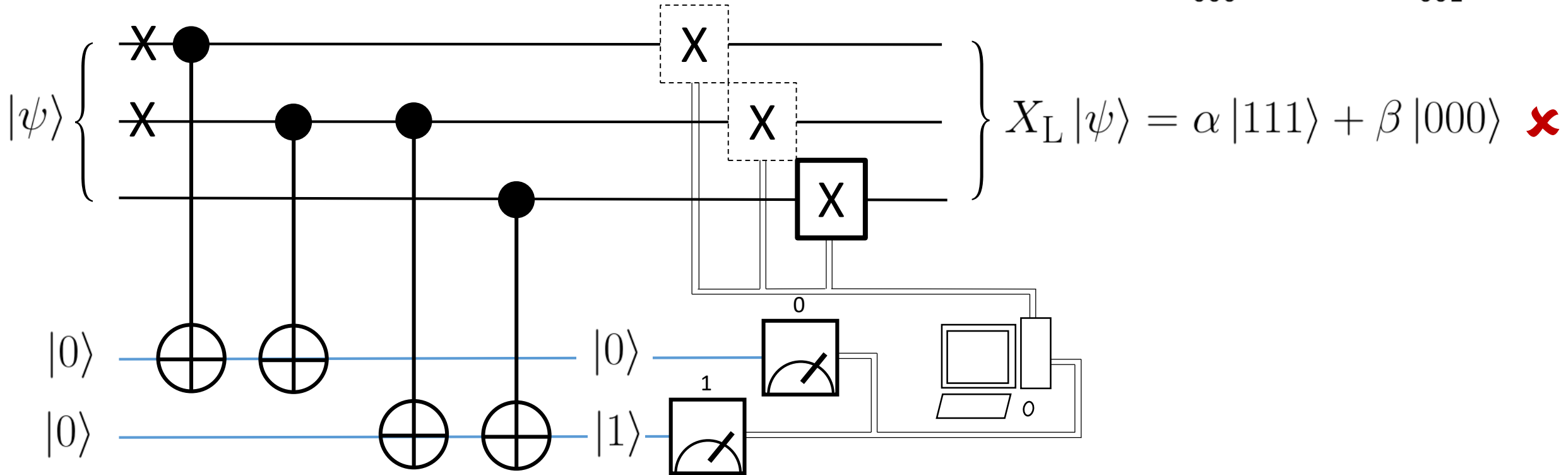
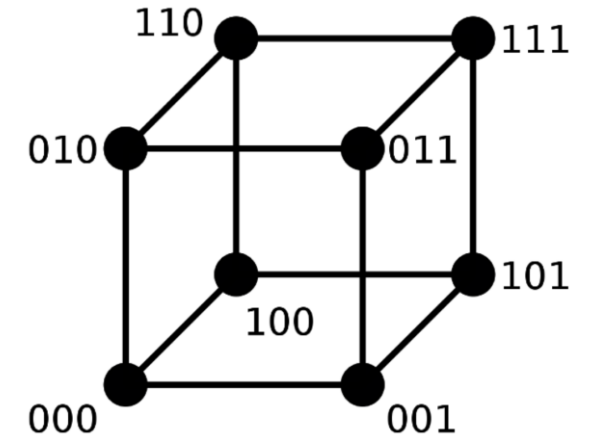


Fault Tolerance

Bit flip Code (Two bit flips)

$$|\tilde{\psi}\rangle = \alpha |110\rangle + \beta |001\rangle$$

Code distance $d = 3$
Only corrects bit flips.

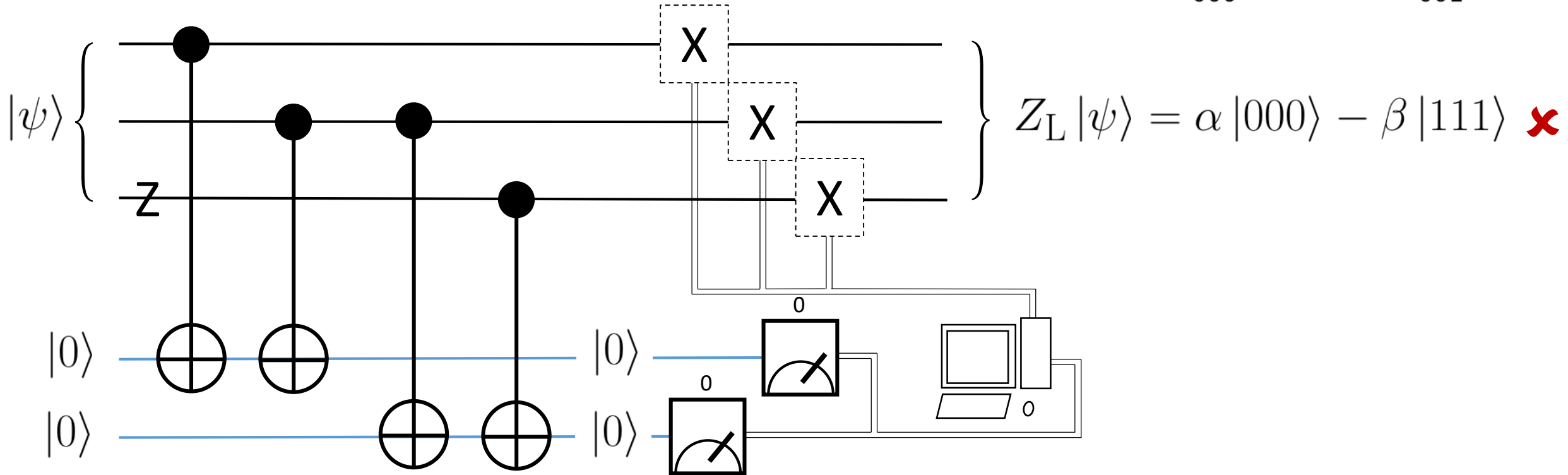
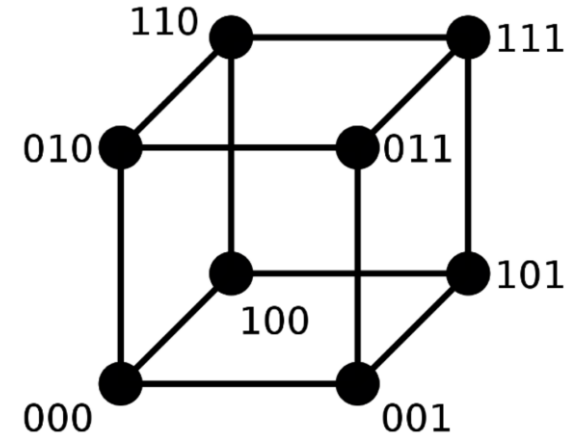


Fault Tolerance

Bit flip Code (One phase flip)

Code distance $d = 3$
Only corrects bit flips.

$$|\tilde{\psi}\rangle = \alpha |000\rangle - \beta |111\rangle$$



Fault Tolerance

Pauli Group and Stabiliser Operators

The Pauli group \mathcal{P}_N for N qubits provides a basis for quantum states:

$$\mathcal{P}_N = \{\pm I, iI, X, Y, Z\}^{\otimes N}$$

An operator S stabilises a state $|\psi\rangle$ if $S|\psi\rangle = |\psi\rangle$, i.e. if it is a symmetry of the state.

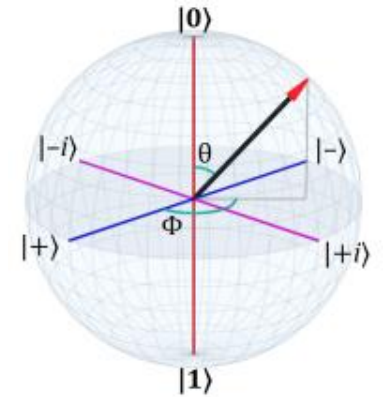
An subgroup \mathcal{S} of \mathcal{P}_N is a stabiliser group if it is *abelian*, namely $[S_i, S_j] = 0 \ \forall S_i, S_j \in \mathcal{S}$ and it does not contain -1 nor $i1$. Then the subspace \mathcal{C} is stabilised by \mathcal{S}

$$\mathcal{C} = \{|\psi\rangle \text{ s.t. } S_i |\psi\rangle = |\psi\rangle \ \forall S_i \in \mathcal{S}\}$$

$$\dim[\mathcal{C}] = 2^{N-m}, \text{ where } m \text{ is the number of generators of } \mathcal{S}.$$

This formalism allows for defining QECC in a compact manner.

Bloch Sphere



$$\begin{aligned} X |+\rangle &= |-\rangle \\ Y |+\rangle &= |i\rangle \\ Z |0\rangle &= |1\rangle \end{aligned}$$

Fault Tolerance

Bit flip Code as Stabiliser Code

$$\mathcal{S} = \{Z_1Z_2, \quad Z_2Z_3\}$$

$|000\rangle$

$|001\rangle$

$|010\rangle$

$|011\rangle$

$|100\rangle$

$|101\rangle$

$|110\rangle$

$|111\rangle$

Fault Tolerance

Bit flip Code as Stabiliser Code

$$\mathcal{S} = \{Z_1 Z_2, \quad Z_2 Z_3\}$$

$$Z_1 Z_2 |000\rangle = + |000\rangle$$

$$Z_1 Z_2 |001\rangle = + |001\rangle$$

$$Z_1 Z_2 |010\rangle = - |010\rangle$$

$$Z_1 Z_2 |011\rangle = - |011\rangle$$

$$Z_1 Z_2 |100\rangle = - |100\rangle$$

$$Z_1 Z_2 |101\rangle = - |101\rangle$$

$$Z_1 Z_2 |110\rangle = + |110\rangle$$

$$Z_1 Z_2 |111\rangle = + |111\rangle$$

Fault Tolerance

Bit flip Code as Stabiliser Code

$$\mathcal{S} = \{Z_1 Z_2, \quad Z_2 Z_3\}$$

$$Z_1 Z_2 |000\rangle = + |000\rangle$$

$$Z_1 Z_2 |001\rangle = + |001\rangle$$

$$Z_1 Z_2 |110\rangle = + |110\rangle$$

$$Z_1 Z_2 |111\rangle = + |111\rangle$$

Fault Tolerance

Bit flip Code as Stabiliser Code

$$\mathcal{S} = \{Z_1 Z_2, \quad Z_2 Z_3\}$$

$$Z_2 Z_3 |000\rangle = + |000\rangle$$

$$Z_2 Z_3 |001\rangle = - |001\rangle$$

$$Z_2 Z_3 |110\rangle = - |110\rangle$$

$$Z_2 Z_3 |111\rangle = + |111\rangle$$

Fault Tolerance

Bit flip Code as Stabiliser Code

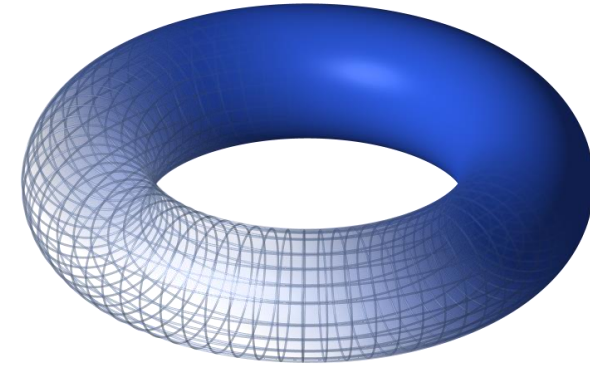
$$\mathcal{S} = \{Z_1 Z_2, \quad Z_2 Z_3\}$$

$$|000\rangle$$

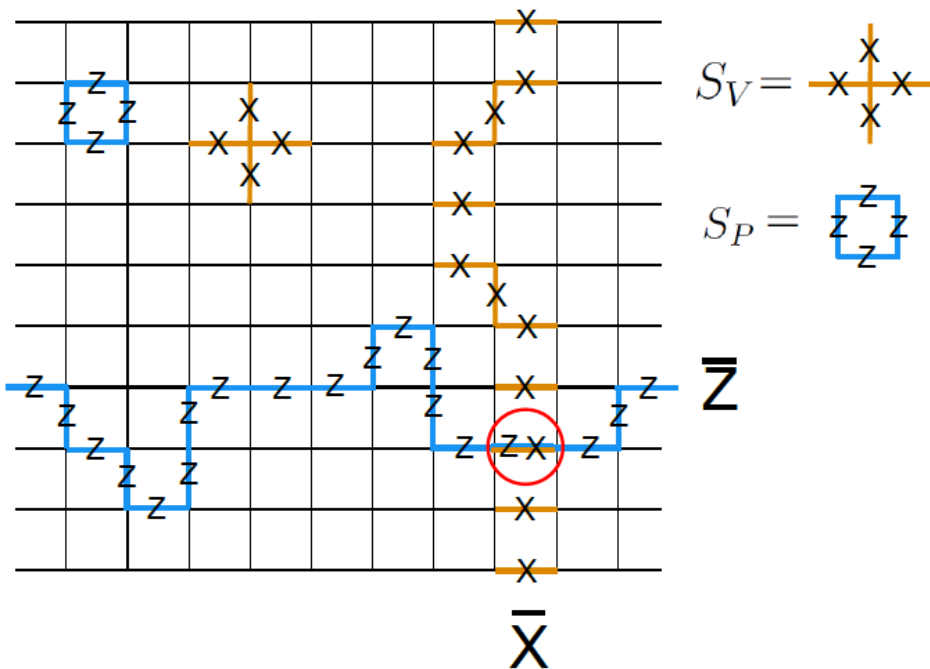
$$|111\rangle$$

Fault Tolerance

Surface Code I: definition



The surface code is defined on a square lattice, on each of whose edges sits a qubit. By introducing periodic boundary conditions, we remove two constraints and a 4-fold degeneracy arises in the ground state.



$$S_V = \prod_{i \in V} X_i$$

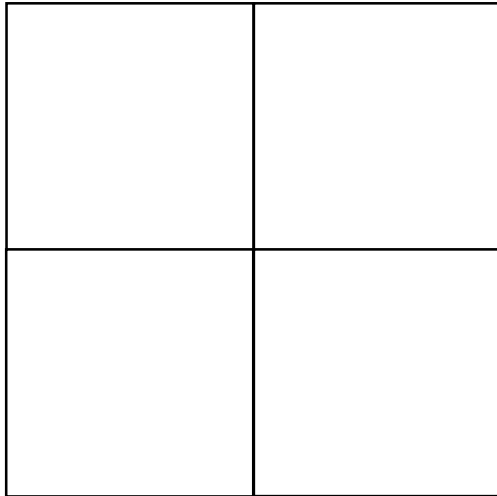
$$S_P = \prod_{i \in P} Z_i$$

$$[S_P, S_V] = 0 \quad \forall P, V$$

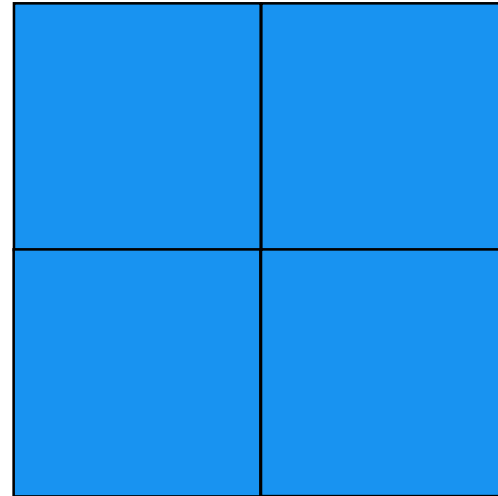
Fault Tolerance

Surface Code I: Example

The surface code is defined on a square lattice, on each of whose edges sits a qubit. By introducing periodic boundary conditions, we remove two constraints and a 4-fold degeneracy arises in the ground state. 4 vertices, 4 plaquettes, and 8 qubits. But only 3 vertices and 3 plaquettes are needed to generate the set. So we have $2^{8-6}=2^2$ dimensional space, so two logical qubits.



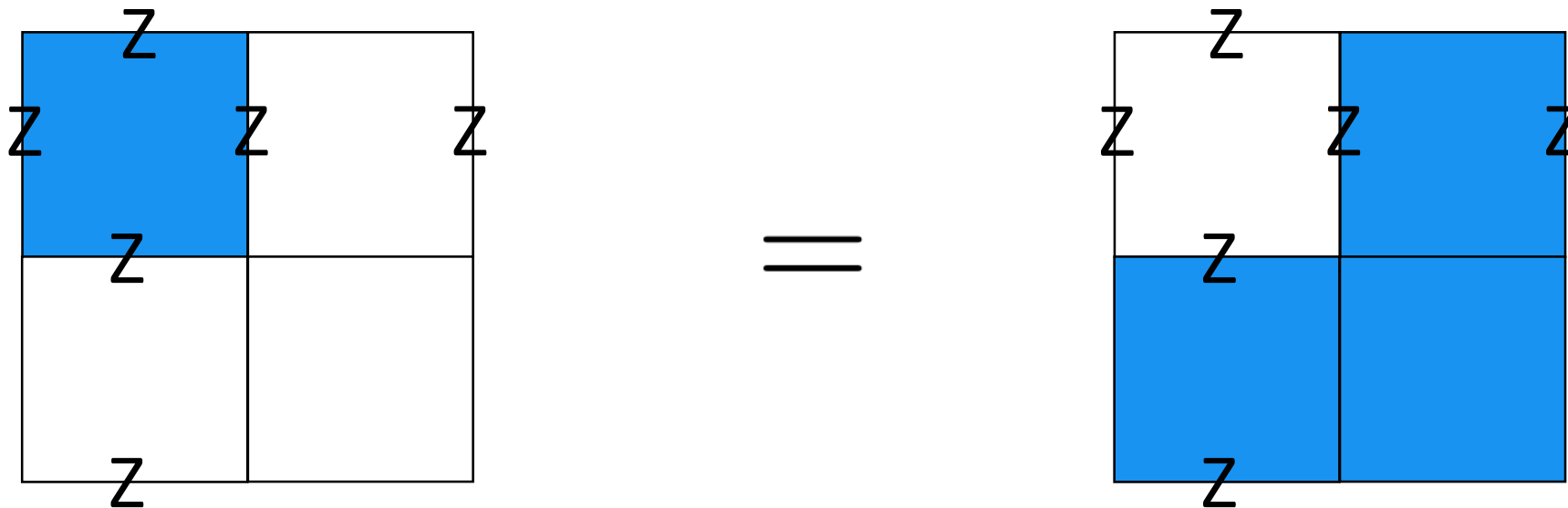
=



Fault Tolerance

Surface Code I: Example

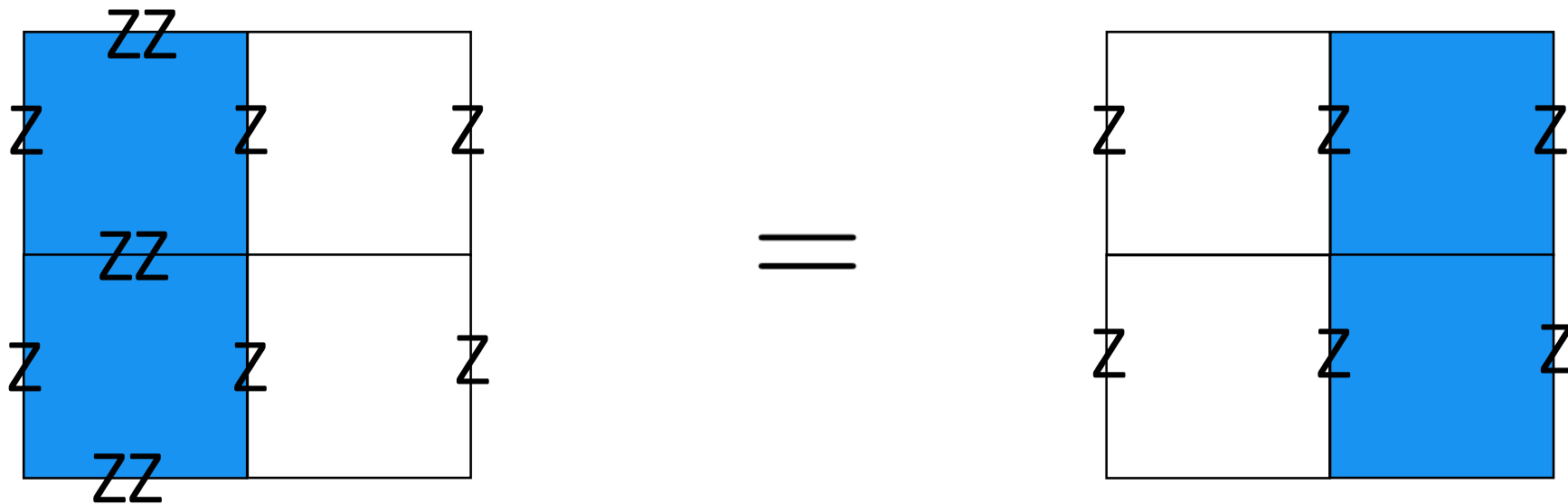
The surface code is defined on a square lattice, on each of whose edges sits a qubit. By introducing periodic boundary conditions, we remove two constraints and a 4-fold degeneracy arises in the ground state. 4 vertices, 4 plaquettes, and 8 qubits. But only 3 vertices and 3 plaquettes are needed to generate the set. So we have $2^{8-6}=2^2$ dimensional space, so two logical qubits.



Fault Tolerance

Surface Code I: Example

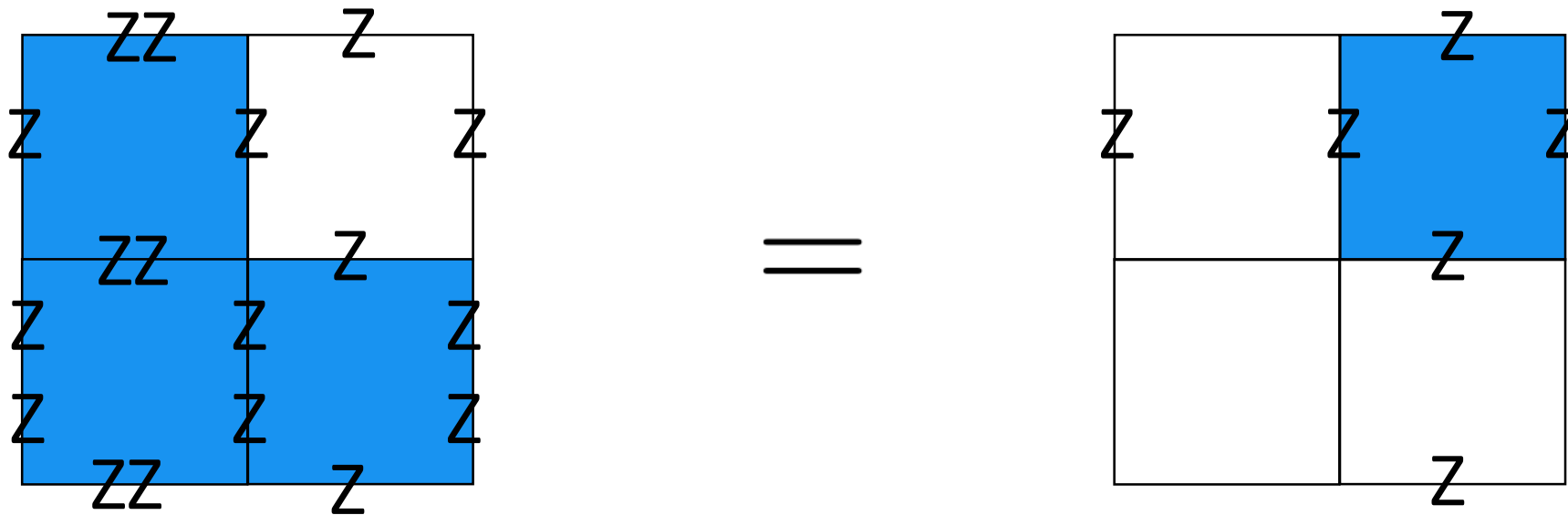
The surface code is defined on a square lattice, on each of whose edges sits a qubit. By introducing periodic boundary conditions, we remove two constraints and a 4-fold degeneracy arises in the ground state. 4 vertices, 4 plaquettes, and 8 qubits. But only 3 vertices and 3 plaquettes are needed to generate the set. So we have $2^{8-6}=2^2$ dimensional space, so two logical qubits.



Fault Tolerance

Surface Code I: Example

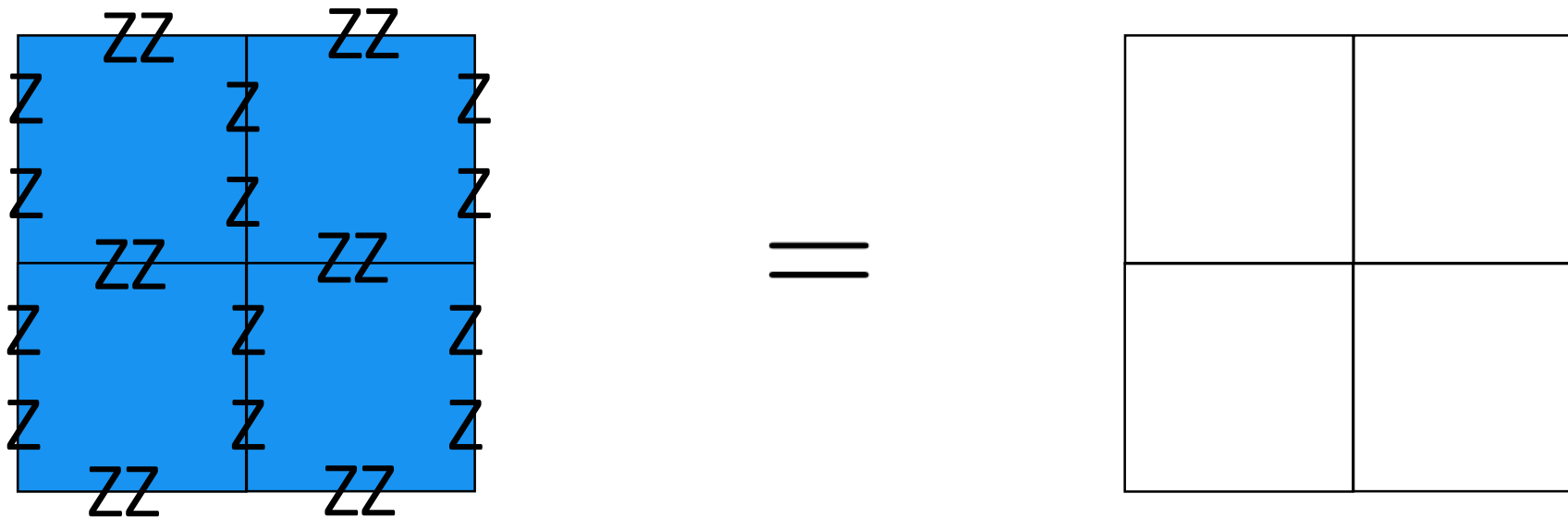
The surface code is defined on a square lattice, on each of whose edges sits a qubit. By introducing periodic boundary conditions, we remove two constraints and a 4-fold degeneracy arises in the ground state. 4 vertices, 4 plaquettes, and 8 qubits. But only 3 vertices and 3 plaquettes are needed to generate the set. So we have $2^{8-6}=2^2$ dimensional space, so two logical qubits.



Fault Tolerance

Surface Code I: Example

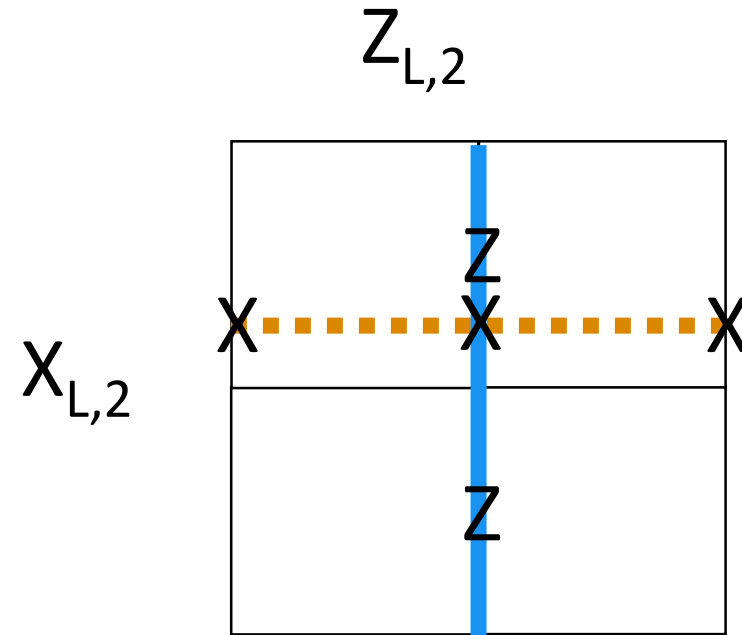
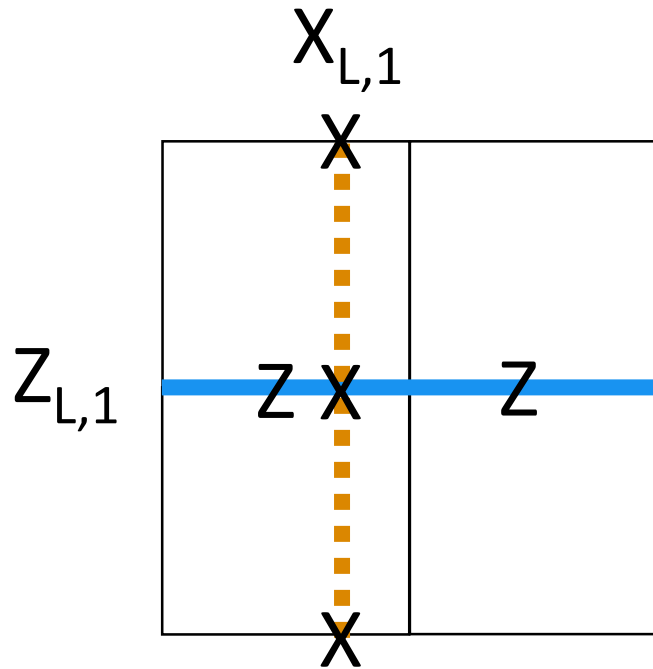
The surface code is defined on a square lattice, on each of whose edges sits a qubit. By introducing periodic boundary conditions, we remove two constraints and a 4-fold degeneracy arises in the ground state. 4 vertices, 4 plaquettes, and 8 qubits. But only 3 vertices and 3 plaquettes are needed to generate the set. So we have $2^{8-6}=2^2$ dimensional space, so two logical qubits.



Fault Tolerance

Surface Code I: Example

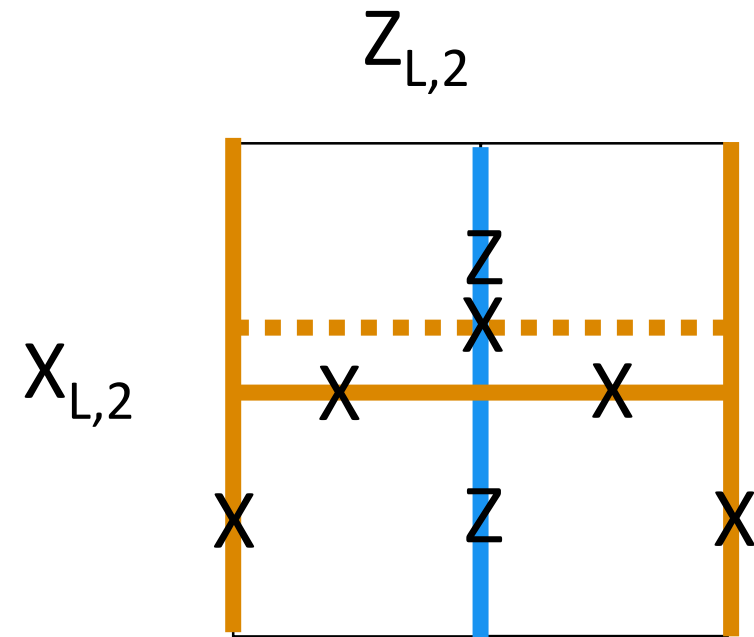
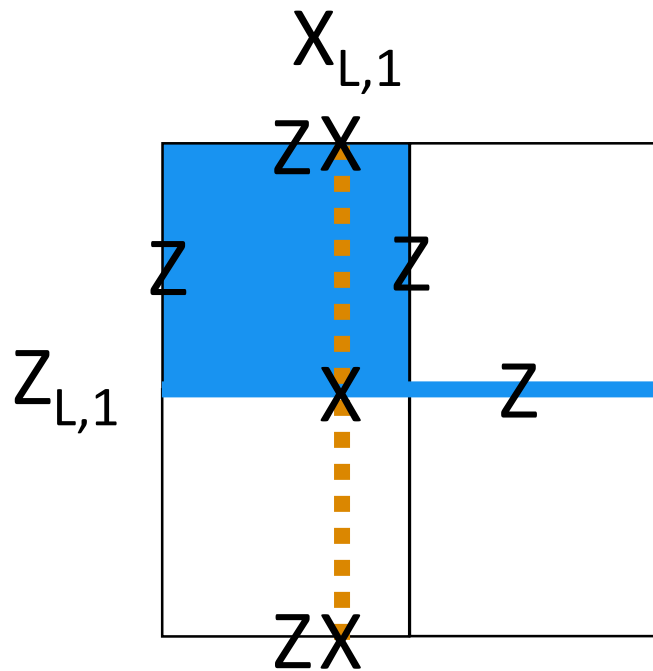
The surface code is defined on a square lattice, on each of whose edges sits a qubit. By introducing periodic boundary conditions, we remove two constraints and a 4-fold degeneracy arises in the ground state. Logical operators commute with plaquettes and vertices.



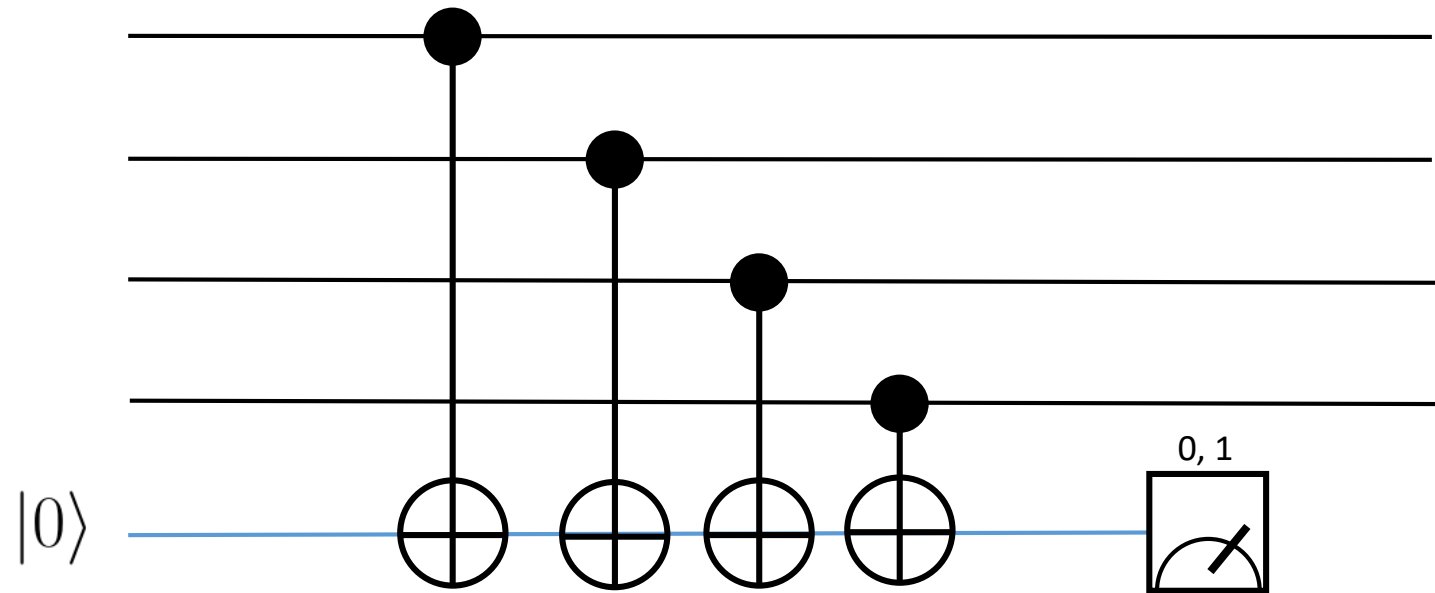
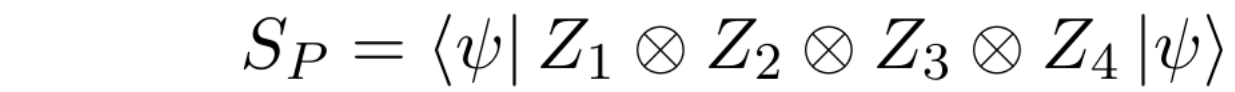
Fault Tolerance

Surface Code I: Example

The surface code is defined on a square lattice, on each of whose edges sits a qubit. By introducing periodic boundary conditions, we remove two constraints and a 4-fold degeneracy arises in the ground state. Logical operators commute with plaquettes and vertices.

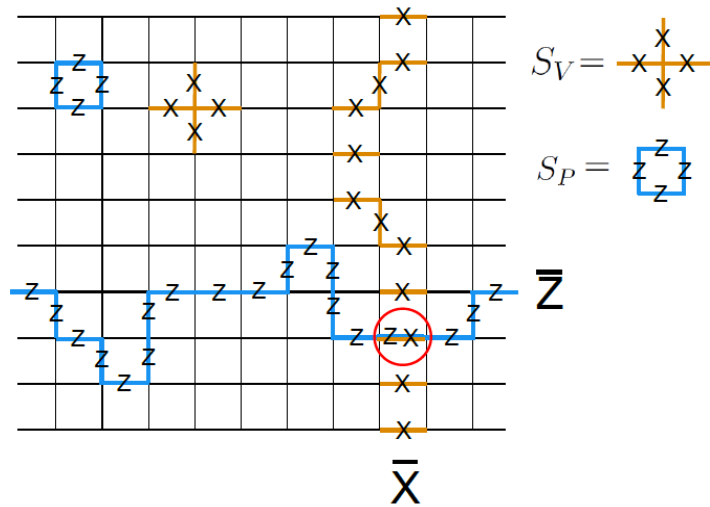


Surface Code I: Syndrome Detection

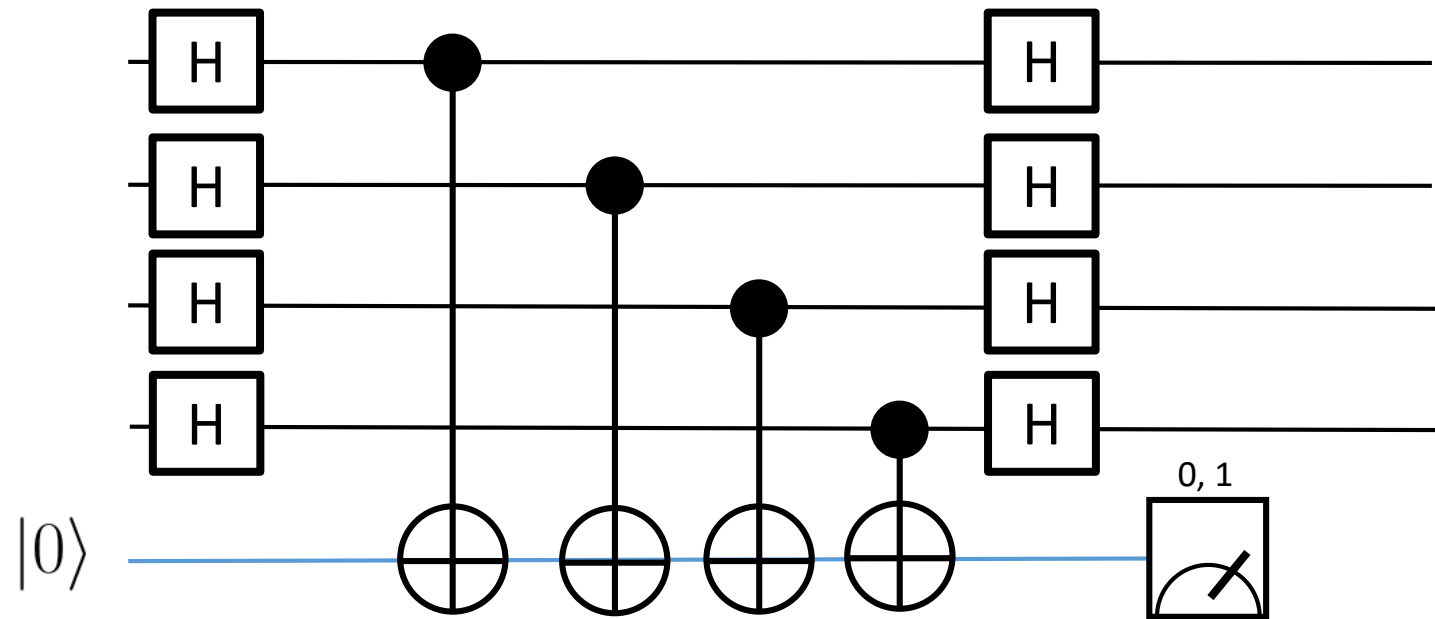


Fault Tolerance

Surface Code I: Syndrome Detection



$$S_V = \langle \psi | X_1 \otimes X_2 \otimes X_3 \otimes X_4 | \psi \rangle$$



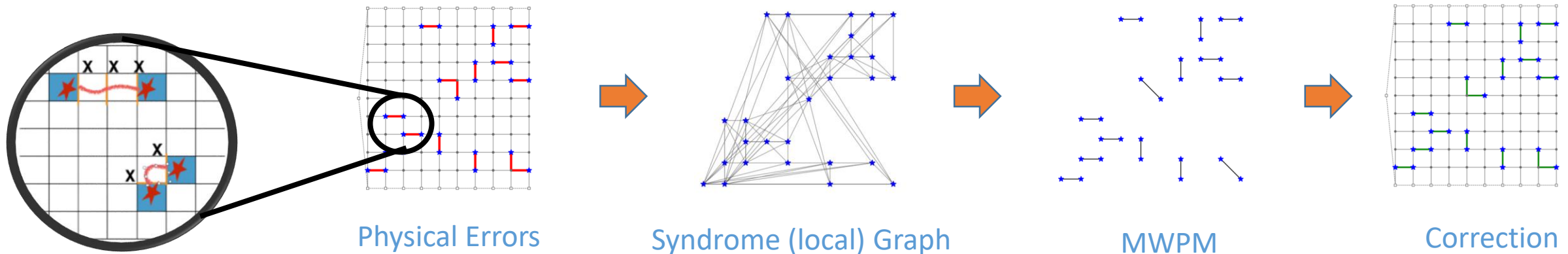
Fault Tolerance

Surface Code II: decoder and threshold

Measuring $\langle S_j \rangle = \langle \psi | E S_j E | \psi \rangle = -\langle \psi | S_j | \psi \rangle$ will give us information about the nature of the errors that potentially occurred, which is the same principle behind the classical parity check matrix.

Once a pattern of flipped plaquette and vertex operators has been obtained, a **distribution of chain endpoints** of errors can be constructed, and an algorithm. A **logical error** occurs when a chain of errors ends up in two boundaries or winds around one hole so that the chain still commutes with the stabilizer, yet is not part of it, thus affecting the logical information.

To initialise in the logical 0 state ($|0\rangle_L$) put all the physical qubits in 0 states and then measure all the stabiliser operators and apply corrections. This works because the logical Z commutes with the stabiliser operators and if all qubits are in 0 states they are in eigenstates of the logical Z operators.



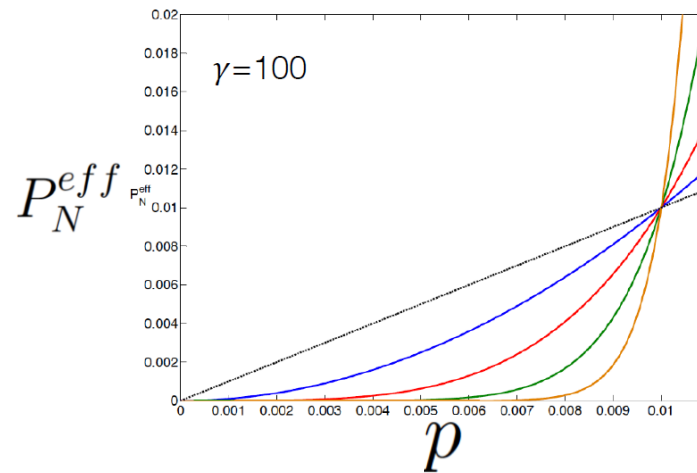
Minimum Weight Perfect Matching (MWPM) decoding is standard for Pauli errors in topological codes. This way of decoding gives a **threshold of $< 1\%$** \rightarrow If gates have an error rate less than the threshold, it is possible to stabilise the computation using the Surface code. We must run the error correction protocol often enough that we prevent sufficient physical errors occurring to induce a logical error, but not too frequently or the errors induced by the correction may become significant.

Fault Tolerance

Fault Tolerance Threshold

The main focus of fault-tolerance is how to use noisy components, as will unavoidably be the case, to **simulate an error-free computation** at the logical level. Whether this is at all possible is a non-trivial question, and the **Threshold Theorem** for quantum computation answers it positively.

$$P_N^{eff} = \frac{(\gamma p)^{2^N}}{\gamma}$$



*Threshold Theorem: it is possible to create a quantum computer to perform an arbitrary quantum computation provided the **error rate per physical gate or time step is below some constant threshold value***

The existence of a threshold for a particular architecture is a **constructive proof for particular QECCs**. Some architectures may not have a threshold, for instance because the physical error rate has some slight dependence on the code distance. This can still be useful.

Fault Tolerance

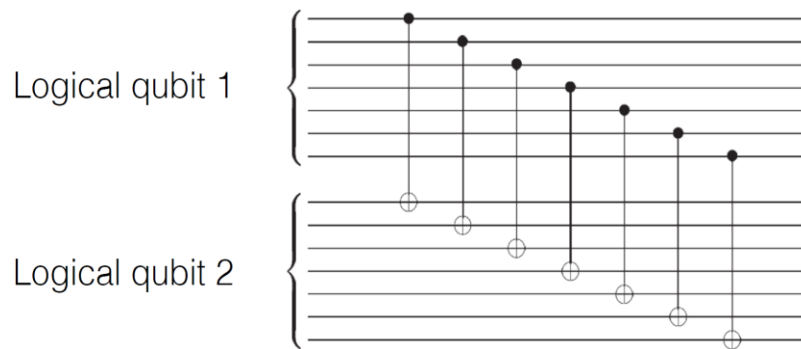
Clifford Operations and Transversality

The Clifford group \mathcal{N}_N is the **normaliser** of the **Pauli group** \mathcal{P}_N , that is, its action leave it invariant modulo conjugation:

$$\mathcal{N}_N = \{S \in U(2^N) | SPS^\dagger \in \mathcal{P}_N, \forall P \in \mathcal{P}_N\}$$

It is important for two reasons:

- 1) circuits made of Clifford gates are classically simulatable
- 2) because of transversality (↓↓↓)



To ensure that logical qubits undergo logical gates while satisfying the requirement that one physical error cascades into at most one local error on each logical qubit one needs to use **transversal gates**.

Transversal gates can be done in a **qubit-wise fashion** ($O_L = \bigotimes_k O_k$), in such a way that an error on the qubit i will only ever propagate to the qubit i on the other block, leaving all other qubits invariant.

→ On stabiliser codes, **Clifford gates can be enacted in a (near) transversal fashion**

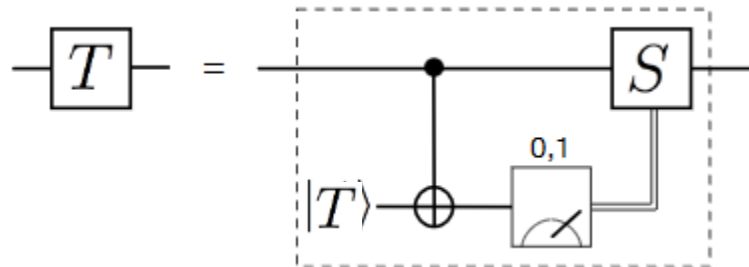
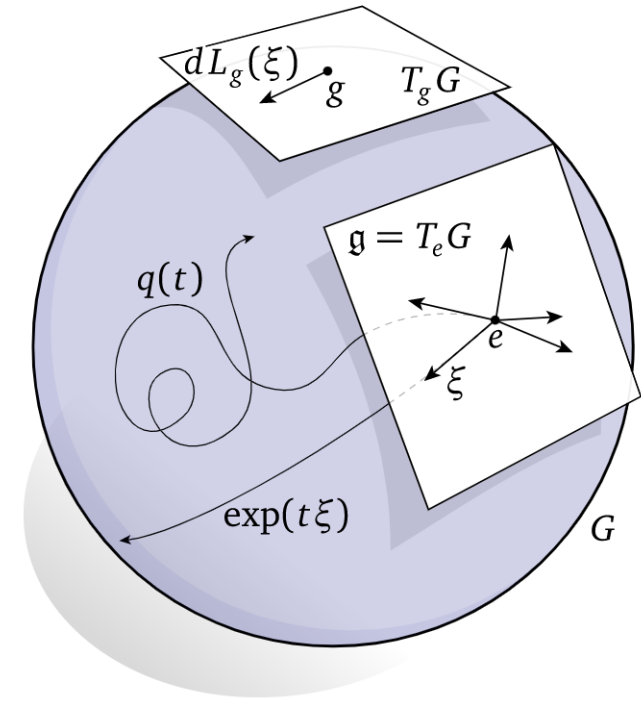
Fault Tolerance

Universality

Ultimately we want to use a quantum computer to approximate unitary transformations acting on quantum states.

Question: *How to reach all of $SU(2^N)$ with a discrete and finite gateset?*

Hint: For $SU(2)$ it typically suffices to choose two 1-qubit operators A and B such that $[A,B] \neq 0$, i.e. they generate a dense subset of $SU(2)$. This can be extended to $SU(2^N)$ using 2-qubit gates (for instance CNOT).



The **Eastin-Knill Theorem** prevents transversal operators O_L , i.e. operators that can be made of single qubit operators $O_L = \bigotimes_k O_k$, to span the whole logical subspace.

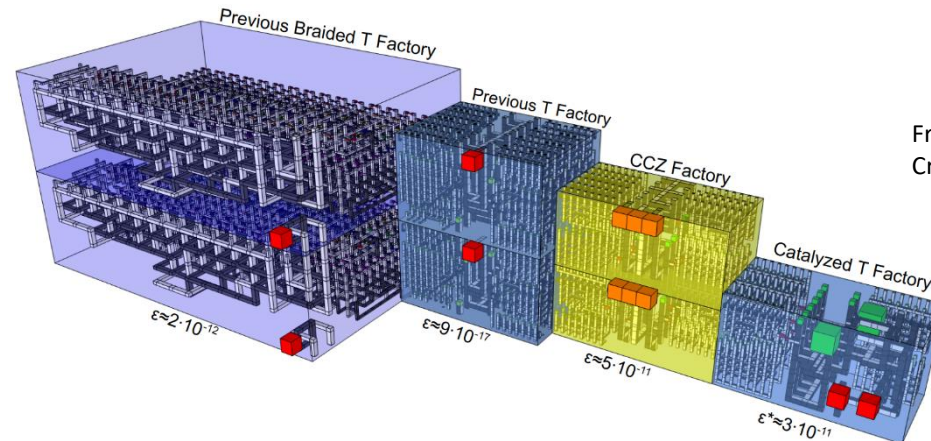
One solution: “**Magic States**” are states that can be obtained from noisy ensembles using Clifford + T. This is a very heavy procedure, but it is the best one that we know to date.

Fault Tolerance

Surface Code III: universality

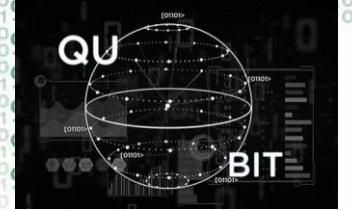
Surface codes can implement CNOT gates transversally between logical qubits. With some extra overhead it can also implement all gates in the Clifford group.

However, in order to attain universal QC, it is necessary to fault-tolerantly create magic states from noisy reservoirs. The computer parts devoted to this procedure are known as **T-factories**.



From [Efficient magic state factories with a catalyzed |CCZ⟩ to 2|T⟩ transformation](#)
Craig Gidney and Austin G. Fowler. Quantum Journal (2019)

These T-factories are extremely resource consuming, as a fault-tolerant decoding algorithm needs to be done. Current estimates place the overhead in the ballpark of 10^5 qubits. **This will be relevant for estimate of runtime of the QPE algorithm for quantum chemistry.**



Quantum Phase Estimation

Recap of DFT and FFT

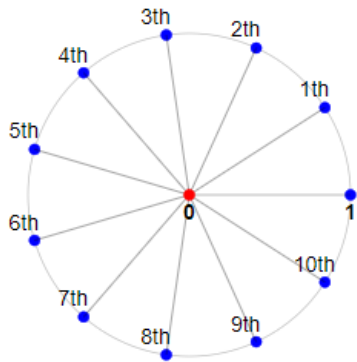
Quantum Fourier Transform

QPE. Error Analysis. Accuracy considerations

Quantum Phase Estimation

Recap of DFT and FFT

Given a vector x of dimension 2^N , the Discrete Fourier Transform of x is given by the 2^N -point multiplication $y = Wx$. The elements of the DFT are powers of the **roots of unity** $\omega = \exp(-2\pi i / 2^N)$



$$\begin{array}{c}
 \xrightarrow{j} \boxed{y = W x} \\
 \\
 W = \frac{1}{\sqrt{2^N}} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_N & \omega_N^2 & \dots & \omega_N^{2^N-1} \\ 1 & \omega_N^2 & \omega_N^4 & \dots & \omega_N^{2(2^N-1)} \\ 1 & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_N^{2^N-1} & \omega_N^{2(2^N-1)} & \dots & \omega_N^{(2^N-1)(2^N-1)} \end{bmatrix} \downarrow k \\
 \\
 y_k = \sum_{j=0}^{2^N-1} x_j e^{-2i\pi jk/2^N}
 \end{array}$$

Quantum Phase Estimation

Recap of DFT and FFT

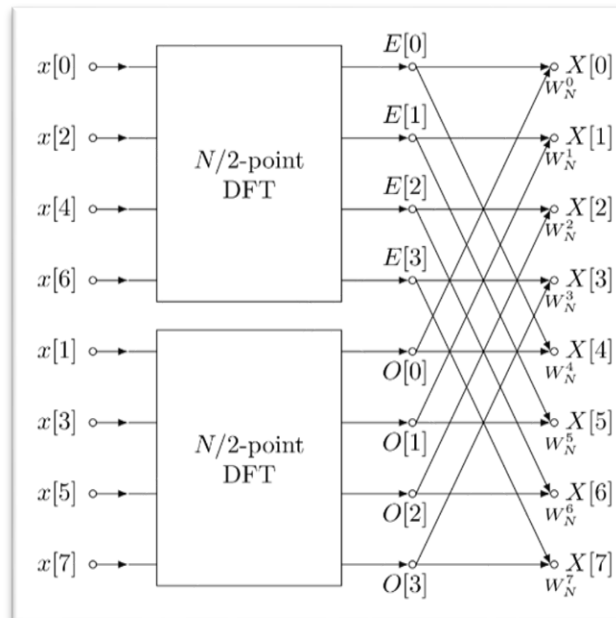
Naively, matrix-vector multiplication takes $O(2^{2N})$ steps.

The **Fast Fourier Transform** algorithm reduces this to $O(N2^N)$, which renders DFT computable in *linear time (in the DFT dimension)*. This is done by **exploiting symmetry** of the DFT matrix representation

However, this is still exponential in N ...

$$\begin{array}{|c|} \hline k \\ \hline \omega^{jk} \\ \hline \end{array} \begin{array}{c} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \vdots \\ \alpha_{n-1} \end{array} = \begin{array}{|c|c|} \hline 2k & 2k+1 \\ \hline \omega^{2jk} & \omega^j \omega^{2jk} \\ \hline \end{array} \begin{array}{c} \alpha_0 \\ \alpha_2 \\ \vdots \\ \alpha_{n-2} \\ \hline \alpha_1 \\ \alpha_3 \\ \vdots \\ \alpha_{n-1} \end{array}$$

even columns odd columns



Principal Discoveries of Efficient Methods of Computing the DFT

Researcher(s)	Date	Lengths of Sequence	Number of DFT Values	Application
C. F. GAUSS [10]	1805	Any composite integer	All	Interpolation of orbits of celestial bodies
F. CARLINI [28]	1828	12	7	Harmonic analysis of barometric pressure variations
A. SMITH [25]	1846	4, 8, 16, 32	5 or 9	Correcting deviations in compasses on ships
J. D. EVERETT [23]	1860	12	5	Modeling underground temperature deviations
C. RUNGE [7]	1903	$2^n K$	All	Harmonic analysis of functions
K. STUMPF [16]	1939	$2^n K, 3^n K$	All	Harmonic analysis of functions
DANIELSON & LANCZOS [5]	1942	2^n	All	X-ray diffraction in crystals
L. H. THOMAS [13]	1948	Any integer with relatively prime factors	All	Harmonic analysis of functions
I. J. GOOD [3]	1958	Any integer with relatively prime factors	All	Harmonic analysis of functions
COOLEY & TUKEY [1]	1965	Any composite integer	All	Harmonic analysis of functions
S. WINOGRAD [14]	1976	Any integer with relatively prime factors	All	Use of complexity theory for harmonic analysis

Quantum Phase Estimation

Quantum Fourier Transform

In ket notation, a state of N qubits corresponds to a vector $|x\rangle$ of dimension 2^N with components x_j .

Take the computational basis:

$$\begin{aligned}
 &|00\dots 00\rangle, |10\dots 00\rangle, |01\dots 00\rangle \dots |11\dots 11\rangle \\
 &\text{i.e.} \\
 &|0\rangle, |1\rangle, |2\rangle \dots |2^N - 1\rangle
 \end{aligned}
 \qquad
 |x\rangle = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{2^N-1} \end{pmatrix} \cdot \begin{pmatrix} |00\dots 00\rangle \\ |10\dots 00\rangle \\ \vdots \\ |11\dots 11\rangle \end{pmatrix}$$

The Fourier transform of state $|x\rangle$ is a state $|y\rangle$, i.e. vector of 2^N components y_k

$$|y\rangle = \sum_{k=0}^{2^N-1} y_k |k\rangle = \sum_{k=0}^{2^N-1} \sum_{j=0}^{2^N-1} e^{2i\pi jk/2^N} x_j |k\rangle$$

Quantum Phase Estimation

QFT: one qubit case

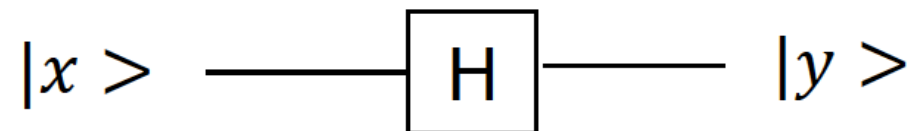
Consider the case where $N = 1$

$$|x\rangle = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \begin{matrix} |0\rangle \\ |1\rangle \end{matrix}$$

$$\omega_1 = e^{i\pi} = -1$$

$$W = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Then the physical dynamics leading to a DFT is a Hadamard gate



Quantum Phase Estimation

QFT: exploiting the symmetry

Note: adding a qubit to a system doubles the size of its Hilbert space. This fact is critical to understanding how the symmetry can be exploited in a way similar to Cooley-Tuckey FFT

By **linearity of QM**, it will suffice to understand how the QFT acts upon a computational basis vector.

$$|j\rangle = |j_0 \dots j_{N-2} j_{N-1}\rangle, j_i \in \{0,1\}$$

$$\begin{aligned} |j\rangle &\xrightarrow{QFT_N} \sum_{k=0}^{2^N-1} e^{2i\pi k j / 2^N} |k\rangle \\ &= \sum_{k=0}^{2^N-1} \omega_N^{jk} |k_0 \dots k_{N-2} k_{N-1}\rangle \end{aligned}$$

$$|k\rangle = |k_0 \dots k_{N-2} k_{N-1}\rangle$$

$$\omega_N = e^{2i\pi/2^N}$$

Quantum Phase Estimation

QFT: exploiting the symmetry

By factoring out the first (or last, depending on encoding) qubit the symmetry (odd – even row) becomes apparent

$$\begin{aligned}
 & \underbrace{\sum_{k=0}^{2^N-1} \omega_N^{jk} |k_0 \dots k_{N-2} k_{N-1}\rangle}_{QFT_N} = \sum_{k=0}^{2^N-1} \omega_N^{jk} |k_0\rangle |k_1 \dots k_{N-1}\rangle \\
 & \quad \begin{array}{c} \boxed{k = \sum_{l=0}^{N-1} k_l 2^{-l}} \quad k' = \frac{k - k_0}{2} \quad \omega_N = e^{2i\pi/2^N} \\ \swarrow \quad \searrow \\ \boxed{k_0 = 0 : \text{ k even}} \quad \boxed{k_0 = 1 : \text{ k odd}} \end{array} \\
 & = \sum_{k'=0}^{2^{N-1}-1} \omega_N^{j(2k')} |0\rangle |k'_0 \dots k'_{N-2}\rangle + \sum_{k'=0}^{2^{N-1}-1} \omega_N^{j(2k'+1)} |1\rangle |k'_0 \dots k'_{N-2}\rangle \\
 & = (|0\rangle + \omega_N^j |1\rangle) \underbrace{\left(\sum_{k=0}^{2^{N-1}-1} \omega_{N-1}^{jk} |k_0 \dots k_{N-2}\rangle \right)}_{QFT_{N-1}} = (|0\rangle + \omega_N^j |1\rangle) (|0\rangle + \omega_{N-1}^j |1\rangle) \cdots (|0\rangle + \omega_0^j |1\rangle)
 \end{aligned}$$

Quantum Phase Estimation

QFT: exploiting the symmetry

A 2^N dimensional QFT can be expressed as a controlled operation tensored with a 2^{N-1} -dimensional QFT

$$\sum_{k=0}^{2^N-1} \omega_N^{jk} |k_0 \dots k_{N-2} k_{N-1}\rangle = (|0\rangle + \omega_N^j |1\rangle) \left(\sum_{k'=0}^{2^{N-1}-1} \omega_{N-1}^{jk'} |k'_0 \dots k'_{N-2}\rangle \right)$$

$$= (|0\rangle + \omega_N^j |1\rangle) (|0\rangle + \omega_{N-1}^j |1\rangle) \cdots (|0\rangle + \omega_0^j |1\rangle)$$

Notice that the ω_N^j in $(|0\rangle + \omega_N^j |1\rangle)$ depends on the **integer value j**, i.e. on the **value of all the qubits**.

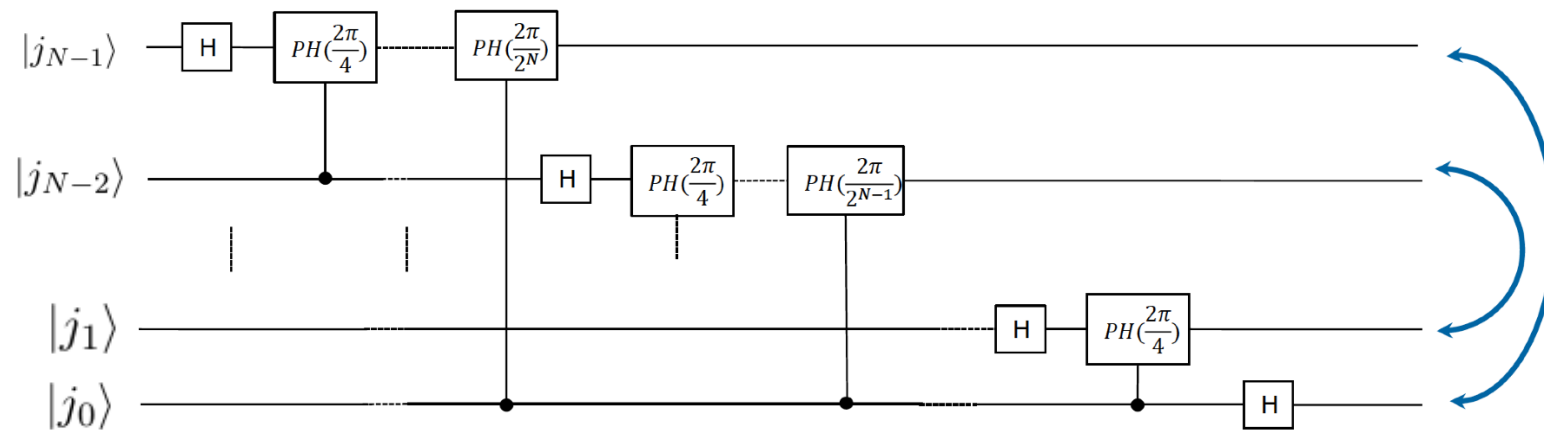
$$j = \sum_{l=0}^{N-1} j_l 2^l \quad \omega_N = e^{2i\pi/2^N}$$

This is implemented by performing controlled rotations on each qubit. The number of controlled rotations is the number of remaining qubits in the QFT.

Quantum Phase Estimation

QFT: runtime analysis

The need for controlled rotations from all-to-one qubit means that one needs around **$N(N-1)/2$ gates** to implement the DFT with a quantum circuit.



The QFT returns the information in reverse ordering of qubits. So a last reversed-ordering step adds $N/2$ swaps gates, each needing 3 CNOTs

The QFT on N qubits has a runtime of $O(N^2)$ while the FFT takes about $O(N2^N)$ steps. So we have shown **an exponential speed-up** in terms of operations and memory.

Quantum Phase Estimation

Goal and assumptions

The goal of the QPE algorithm takes two inputs:

- A unitary operator U
- An eigenstate $|u\rangle$ of U

Since U is unitary, its eigenvalues u have modulus 1 and can be written $u = e^{2\pi i\varphi}$. The QPE algorithm returns the value of φ corresponding to $|u\rangle$ in **binary fraction format** with precision t .

Two important assumptions underlie the QPE routine:

- Being able to perform the **gate U^K controlled by a register of t qubits**, for non-negative K .
- Being able to prepare the state $|u\rangle$ as input for the controlled U^K . (This can be relaxed at the expense of introducing **randomness**)

Quantum Phase Estimation

Binary Fractions

Quantum Phase Estimation operates on a binary fraction encoding.

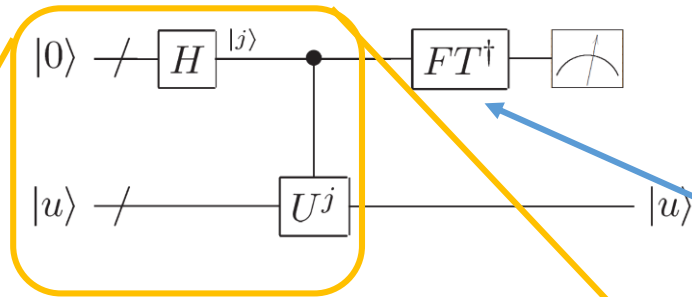
Given a real $\varphi < 1$ expressed in decimal basis, its binary fraction $\varphi = 0.\varphi_0\varphi_1\varphi_2\varphi_3\dots$ can be written as

$$\varphi = \sum_{j=0}^{\infty} \varphi_j 2^{-j-1}$$

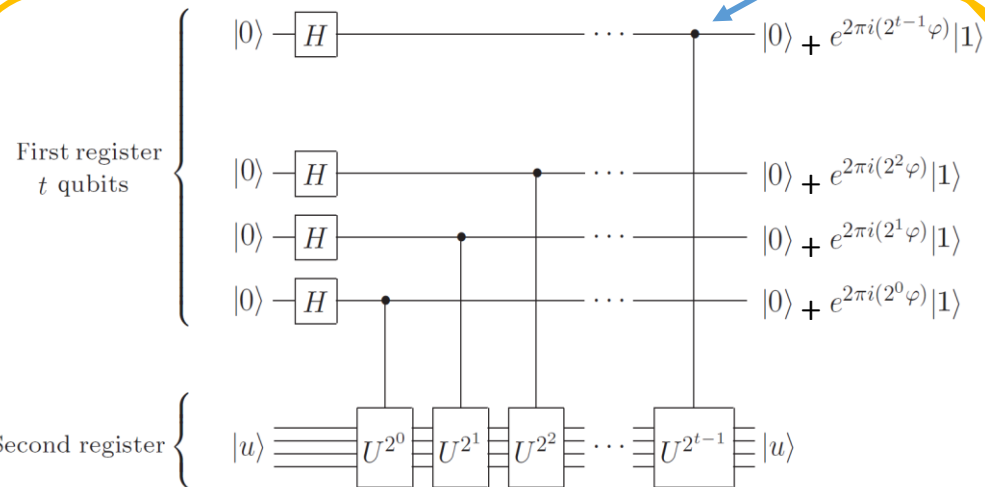
For instance, the number 0.72 in base 10 can be expressed as $0.1011011\dots = 0.7109 \approx 0.72$

The precision of the binary fraction will depend on the number t of bits (or qubits) available.

Quantum Phase Estimation Algorithm



1. $|0\rangle|u\rangle$ initial state
2. $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|u\rangle$ create superposition
3. $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle U^j |u\rangle$ apply black box
 $= \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} e^{2\pi i j \varphi_u} |j\rangle|u\rangle$ result of black box
4. $\rightarrow |\widetilde{\varphi}_u\rangle|u\rangle$ apply inverse Fourier transform
5. $\rightarrow \widetilde{\varphi}_u$ measure first register



The **first register** is necessary to implement the controlled U^k gates. The length of this register **determines the accuracy** of the phase estimation.

The length of the **second register** will depend on the problem under consideration, i.e. the **size of $|u\rangle$**

Quantum Phase Estimation

Success probability

Q1 And what happens if the prepared state is not $|u\rangle$, but some $|u'\rangle = c_{\parallel}|u\rangle + c_{\perp}|u^{\perp}\rangle$?

Q2 What happens if $\varphi = 0$. $\varphi_0\varphi_1\varphi_2\varphi_3\dots$ cannot be expressed exactly using only t qubits?

Q3 How can we mitigate this fact?

Quantum Phase Estimation

Success probability

Q1 And what happens if the prepared state is not $|u\rangle$, but some $|u'\rangle = c_{\parallel}|u\rangle + c_{\perp}|u^{\perp}\rangle$?

A1 *The algorithm works with probability c_{\parallel}^2 :*

$$\begin{aligned} |0\rangle |u'\rangle &\xrightarrow{H+C-U} \\ &\rightarrow \sum_j |j\rangle U^j |u'\rangle = c_{\parallel} \sum_j |j\rangle U^j |u\rangle + c_{\perp} \sum_j |j\rangle U^j |u^{\perp}\rangle \\ &\xrightarrow{iQFT} c_{\parallel} |\tilde{\varphi}_u\rangle |u\rangle + c_{\perp} |\tilde{\varphi}_{u^{\perp}}\rangle |u^{\perp}\rangle \end{aligned}$$

Q2 What happens if $\varphi = 0$. $\varphi_0\varphi_1\varphi_2\varphi_3\dots$ cannot be expressed exactly using only t qubits?

Q3 How can we mitigate this fact?

Quantum Phase Estimation

Success probability

Q1 And what happens if the prepared state is not $|u\rangle$, but some $|u'\rangle = c_{\parallel}|u\rangle + c_{\perp}|u^{\perp}\rangle$?

A1 *The algorithm works with probability c_{\parallel}^2 :*

$$\begin{aligned}
 |0\rangle |u'\rangle &\xrightarrow{H+C-U} \\
 &\rightarrow \sum_j |j\rangle U^j |u'\rangle = c_{\parallel} \sum_j |j\rangle U^j |u\rangle + c_{\perp} \sum_j |j\rangle U^j |u^{\perp}\rangle \\
 &\xrightarrow{iQFT} c_{\parallel} |\tilde{\varphi}_u\rangle |u\rangle + c_{\perp} |\tilde{\varphi}_{u^{\perp}}\rangle |u^{\perp}\rangle
 \end{aligned}$$

Q2 What happens if $\varphi = 0$. $\varphi_0\varphi_1\varphi_2\varphi_3\dots$ cannot be expressed exactly using only t qubits?

A2 *The inverse QFT will not be peaked at $|\tilde{\varphi}\rangle = |2^t \varphi\rangle$ so there will be a bias error.*

Q3 How can we mitigate this fact?

Quantum Phase Estimation

Success probability

Q1 And what happens if the prepared state is not $|u\rangle$, but some $|u'\rangle = c_{\parallel}|u\rangle + c_{\perp}|u^{\perp}\rangle$?

A1 *The algorithm works with probability c_{\parallel}^2 :*

$$\begin{aligned} |0\rangle |u'\rangle &\xrightarrow{H+C-U} \\ &\rightarrow \sum_j |j\rangle U^j |u'\rangle = c_{\parallel} \sum_j |j\rangle U^j |u\rangle + c_{\perp} \sum_j |j\rangle U^j |u^{\perp}\rangle \\ &\xrightarrow{iQFT} c_{\parallel} |\tilde{\varphi}_u\rangle |u\rangle + c_{\perp} |\tilde{\varphi}_{u^{\perp}}\rangle |u^{\perp}\rangle \end{aligned}$$

Q2 What happens if $\varphi = 0$. $\varphi_0\varphi_1\varphi_2\varphi_3\dots$ cannot be expressed exactly using only t qubits?

A2 *The inverse QFT will not be peaked at $|\tilde{\varphi}\rangle = |2^t \varphi\rangle$ so there will be a bias error.*

Q3 How can we mitigate this fact?

A3 *Adding extra qubits increases accuracy exponentially. For $t = n + p$ qubits and target accuracy $\varepsilon_{QPE} = 2^{-n}$ and failure probability δ on the order of $p = O(\log(1/\delta))$ qubits are needed.*

Quantum Phase Estimation

Accuracy considerations

The resources needed to estimate φ to an accuracy $\varepsilon = 2^{-t}$ are ^(*)

- $O(\log(1/\varepsilon))$ qubits
- $O(1/\varepsilon)$ controlled U gates, and
- $O(\log(1/\varepsilon)^2)$ gates in the inverse QFT.

This has to be compared with the #repetitions in a single step of the optimization in hybrid classical-quantum variational algorithms, which scale as $\text{\#reps} \sim O(1/\varepsilon^2)$ because of **statistical sampling**.

This should not be too surprising as **QPE is known to be BQP-complete**, i.e. as hard as the hardest problem solvable in poly time by a QC. However, QPE is much more demanding in terms of circuit depth and gate fidelity. It is not expected to work without error correction.

^(*) Assuming that we can prepare a state with sufficiently high overlap with $|u\rangle$



Limitations of FTQC in Q. Chemistry

QPE for Hamiltonian Simulation

State Preparation

Gate overhead

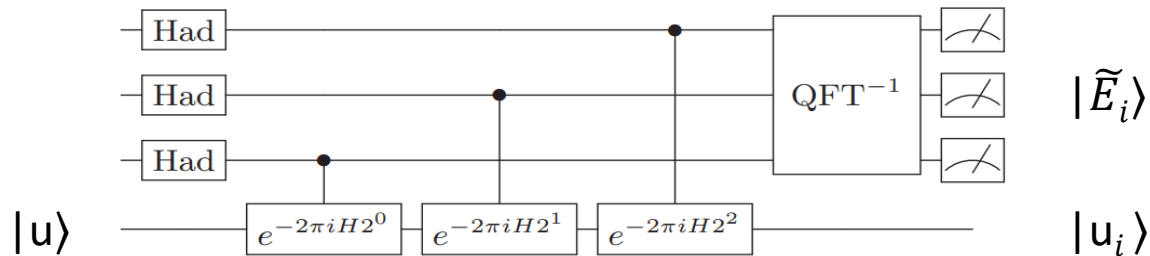
Limitations of QC in Quantum Chemistry

QPE for Chemistry Problems

The QPE can be modified to obtain the ground state energy and eigenstate of a chemical system.

Given a molecular Hamiltonian H , the idea is to use

- $U = \exp(-2\pi i H \tau)$ for $\tau = 1, 2, 4, \dots, 2^t$
- and $|u\rangle$ has non-vanishing overlap with $|E_0\rangle$, i.e. $|u\rangle = \sum_i c_i |E_i\rangle$ and $c_0^2 \leq 1$

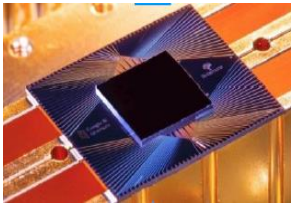


The **total coherent simulation time** is $T \approx 2^{t+1} \pi$ which entails an accuracy $\epsilon = O(1/T)$, as opposed to what can be obtained via statistical sampling $\epsilon = O(1/T^{1/2})$

Limitations of QC in Quantum Chemistry

Qubits to Fermions

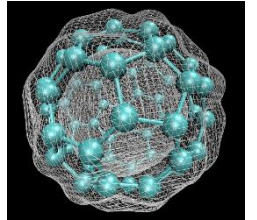
One of the main interests of *digital* QC (opposed to *analog* quantum simulators) is that it allows us to simulate Fermionic systems using two-level systems. In order to express a **electronic Hamiltonian** in a qubit representation, the **fermionic anticommutation** relations need to be artificially enforced from a **Pauli algebra**.



$$\begin{aligned} [P_i, P_j] &= 2i \sum_{k=1}^3 \epsilon_{ijk} P_k \\ \{P_i, P_j\} &= 2\delta_{ij} \end{aligned}$$



$$\begin{aligned} \{a_i, a_j\} &= \{a_i^\dagger, a_j^\dagger\} = 0 \\ \{a_i, a_j^\dagger\} &= \delta_{ij} \end{aligned}$$



The **Jordan-Wigner representation** is the most intuitive one:

$$a_k = Z_0 \otimes Z_1 \otimes \dots \otimes Z_{k-1} \otimes Q_k$$

$$a_k^\dagger = Z_0 \otimes Z_1 \otimes \dots \otimes Z_{k-1} \otimes Q_k^\dagger$$

$$Q_k = (X_k + iY_k)/2$$

$$Q_k^\dagger = (X_k - iY_k)/2$$

Whereas the amount of classical memory needed to store the FCI wavefunction is exponential in M (#spin-orbitals), a quantum computer only needs M qubits → **exponential savings in memory**

Limitations of QC in Quantum Chemistry

Scaling considerations

For M spin-orbitals, using Gaussian basis functions, the number of interaction terms in the Hamiltonian grows as $O(M^4)$, each of which acts on $O(M)$ qubits.

$$H = \sum_{p,q} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{p,q,r,s} h_{pqrs} a_p^\dagger a_q^\dagger a_r a_s \quad \xrightarrow{\text{Jordan-Wigner}} \quad H = \sum_i^{O(M^4)} h_i P_i \quad |P_i| \propto O(M)$$

Now the issue is how to simulate the unitary evolution having access to the Hamiltonian. The simplest/earliest approach relies on **Trotterization** of the dynamics:

$$U = e^{-iHt} = \left(\prod_i e^{-ih_i P_i t/r} \right)^r + O(t^2/r) = U^{\text{Trott}} + O(t^2/r)$$

For a fixed accuracy ϵ_{Trott} , the **gate complexity** is $O(M \times M^4 \times r)$. The explicit calculation of r is difficult.

Limitations of QC in Quantum Chemistry

Bottleneck 1: State Preparation

Given M (#qubits) and target accuracy ϵ , the cost of implementing QPE-based quantum chemistry calculations have three components:

$$C_{\text{QPE}} = \text{poly}(1/S) [\text{poly}(M)\text{poly}(1/\epsilon) + C]$$

1. C is the cost of preparing the initial state $|u\rangle$ (i.e. **mean field**)
2. $\text{poly}(M) \text{poly}(1/\epsilon)$ is the cost of implementing the QPE algorithm
3. $\text{poly}(1/S)$ with $S = |\langle u | E_0 \rangle|^2$ is the expected number of repetitions

Unfortunately, there is frequently a hidden dependency of S on M .

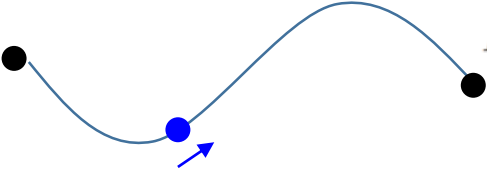
In **worst case scenario**, if the initial state $|u\rangle = |\Psi_1\rangle \dots |\Psi_{O(k)}\rangle$ is a product of $O(k)$ non-interacting components, each of which has an overlap $< s$ with target state, the **#reps scales as $s^{O(k)}$**

Limitations of QC in Quantum Chemistry

Bottleneck 1: State Preparation

State preparation is of critical importance to render QPE-based computation practical.

The idea of **Adiabatic State Preparation** is to start from the ground state of some effective non-interacting Hamiltonian (Hartree-Fock for instance) and progressively turn on the interaction terms between subsystems.

$$H_0 = \sum_{pq} (h_{pq} + V_{pq}^{\text{eff}}) a_p^\dagger a_q$$


$$H = \sum_{p,q} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{p,q,r,s} h_{pqrs} a_p^\dagger a_q^\dagger a_r a_s$$

$$H_s = \left(1 - \frac{s}{T}\right) H_0 + \frac{s}{T} H$$

$$T \approx \mathcal{O}\left(\frac{M^4}{\min_t \Delta^2(t)}\right)$$

- **Gap** The duration T is inversely proportional to the minimum energy gap $\Delta = E_1 - E_0$ between the ground state energy and the first excited state, squared. Only $\min_t \Delta \sim 1/\text{poly}(M)$ scaling is acceptable. **The gap is unknown a priori!**
- **Temperature** Even for favourable scalings, $k_B T_H \geq \min_t \Delta \sim 1/\text{poly}(M)$ for large enough problems. Noise will cause transitions from the ground state into high energy states and decrease success probability.



Limitations of QC in Quantum Chemistry

Bottleneck 2: Gate count

The overarching goal of QC for chemistry is to perform calculations with a precision above **chemical accuracy** $\epsilon_{\text{CH}} = 1\text{kcal / mol} \approx 1.6 \cdot 10^{-3} \text{ Hartree}$

Given a Hamiltonian H , the goal is to simulate the Trotterized unitary U^{Trott} with a quantum circuit \hat{C} satisfying:

1. $\hat{C} = g_1 g_2 \dots g_L$ is *synthesized* using a discrete set of gates $\{g_i\}$ such that

$$\|U^{\text{Trott}} - \hat{C}\| = \|U^{\text{Trott}} - \prod_i^L g_i\| \leq \epsilon_{\text{Synth}}$$

2. All the accumulated errors must lie below chemical accuracy

$$\epsilon_{\text{QPE}} + \epsilon_{\text{Trott}} + \epsilon_{\text{Synth}} \leq \epsilon_{\text{Chemical}}$$

For a fixed number of precision qubits (cf. ϵ_{QPE}) and Trotter step (cf. ϵ_{Trott}), it is possible to **estimate L from ϵ_{Synth}**

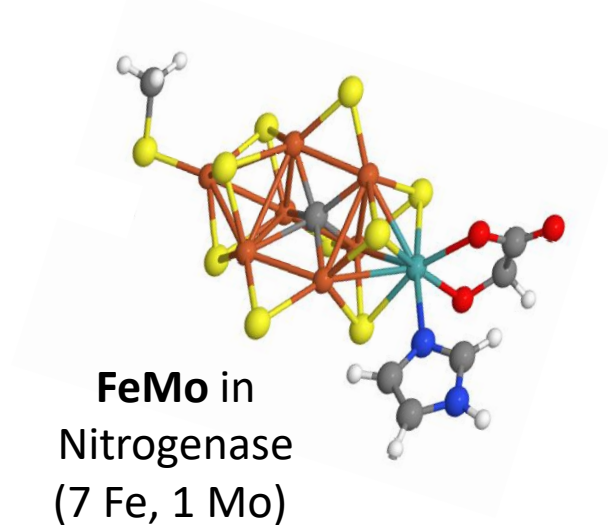
Limitations of QC in Quantum Chemistry

Bottleneck 2: Gate count

The state-of-the-art QECC for achieving fault-tolerance is the **surface code**, in which universality is achieved through the gateset {Clifford + T}

Most **Clifford gates** are transversal in the surface code, which means that they are “**easy**”, in the sense that can be prepared at reasonable time and cost (transversally). **Non-Clifford gates**, such as T, rely on a heavy procedure known as “**state distillation**”, in which many noisy T states undergo several steps of decoding to achieve a clean $|T\rangle$ state that will be used to implement $\pi/8$ rotations at the logical level. ~

Estimates for computing E_0 of FeMoco using around **54 space orbitals**

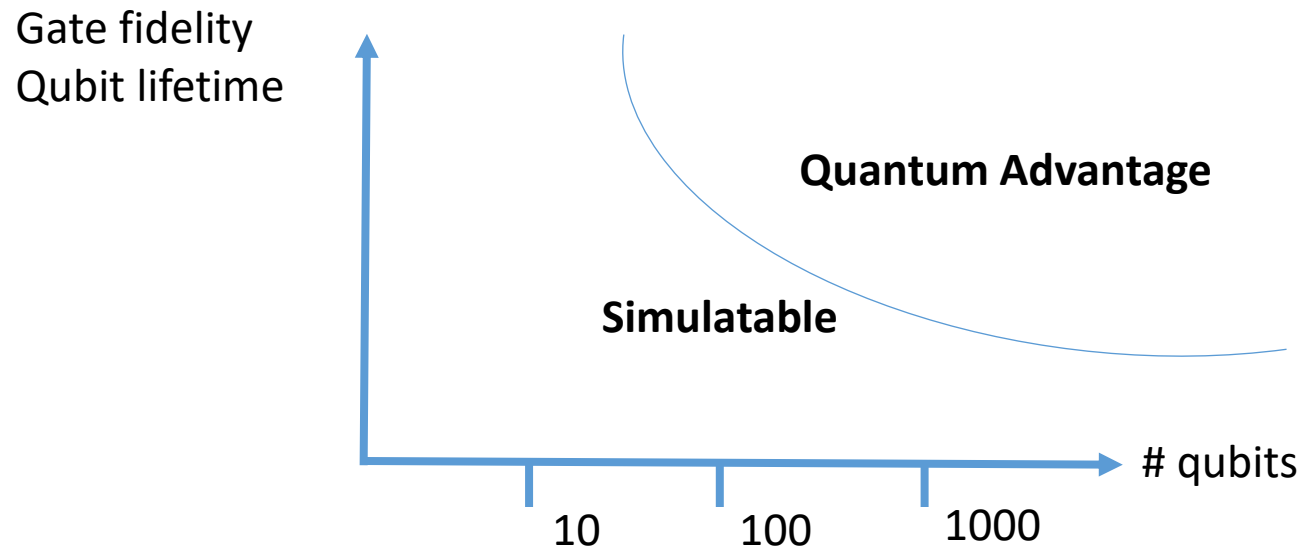


Resources	Estimates
# logical qubits	~110
# phys. qubits / log. qubit	$\sim 10^3 - 10^4$
# T-complexity	$\sim 10^2$
# qubits per T-factory	$\sim 10^5$
Total # qubits	$\sim 10^7 - 10^8$
Expected runtime for QPE	days to months (assuming gate speed < 10 nsec)

Limitations of QC in Quantum Chemistry

Bottleneck 3: quantity vs quality

Gate fidelity and **qubit quality** are the most critical factors to building a quantum computer.



→ **Improving qubit control** and implementing **active noise suppression** is much more important than manufacturing millions of qubits with limited control/gate fidelity.

References

Fault Tolerant Quantum Computation

- Gottesman, D. (2006). Quantum error correction and fault-tolerance. *Quantum Information Processing: From Theory to Experiment*, 199, 159.
- 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.

Quantum Phase Estimation

Nielsen, M. A., & Chuang, I. (2002). *Quantum computation and quantum information*.

Quantum Computing for Chemistry

- Reiher, M., Wiebe, N., Svore, K. M., Wecker, D., & Troyer, M. (2017). Elucidating reaction mechanisms on quantum computers. *Proceedings of the national academy of sciences*, 114(29), 7555-7560.
- McArdle, S., Endo, S., Aspuru-Guzik, A., Benjamin, S. C., & Yuan, X. (2020). Quantum computational chemistry. *Reviews of Modern Physics*, 92(1), 015003.
- Lee, S., Lee, J., Zhai, H., Tong, Y., Dalzell, A. M., Kumar, A., ... & Chan, G. K. (2022). Is there evidence for exponential quantum advantage in quantum chemistry?. *arXiv preprint arXiv:2208.02199*.

Appendices

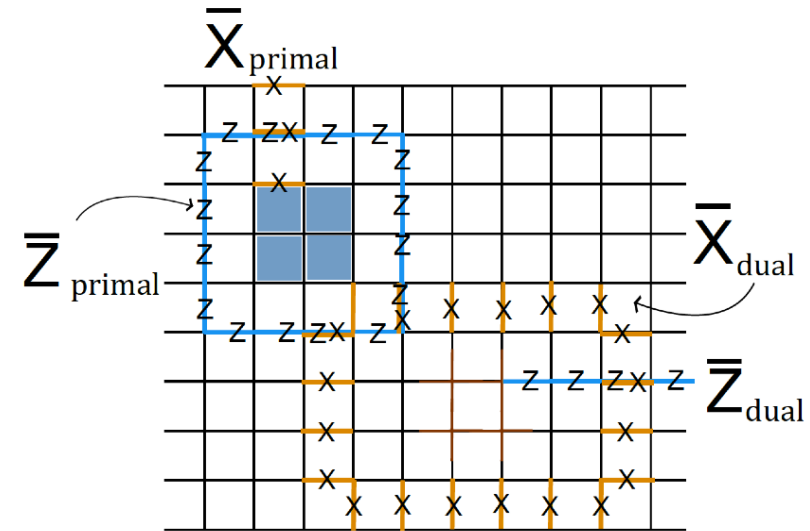
Fault Tolerance

Surface Code I: definition

Surface codes can have two types of boundary: **primal boundaries** are associated to **plaquettes**, whereas **dual boundaries** correspond to **vertices**.

A **primal hole** is created by not enforcing plaquette operators, whereas a **dual hole** arises from not enforcing vertex operators.

They encode primal and dual qubits, respectively.



To attain a non-trivial topology some of the inner stabilizer operators will not be enforced, which can be seen as a poking "**hole**" in the code. Holes can also be primal or dual, depending on which type of stabilizers are released. Not enforcing one stabilizer operator will correspond to releasing a two-dimensional subspace from the constraint imposed by the operator, which is consistent with the idea that logical codewords live in subspaces of the total Hilbert space.

Stabilisers

$$\mathcal{S} = \{g_k\}_{k=1}^m, \quad \mathcal{C} = \{|\psi\rangle \text{ s.t. } g_k |\psi\rangle = |\psi\rangle \quad \forall 1 \leq k \leq m\}$$

$$\mathcal{P}_{\mathcal{S}, \vec{x}} = \prod_{k=1}^m \frac{1 + (-1)^{x_k} g_k}{2}, \quad [g_k, g_l]_- = 0, \quad g_k \neq \pm \mathbb{I}, \quad g_k \neq \pm i\mathbb{I},$$

$$(\vec{x})_k = x_k = \begin{cases} 0 \\ 1 \end{cases}, \quad (\vec{x}_0)_k = 0, \quad \mathcal{P}_{\mathcal{S}, \vec{x}_0} |\psi\rangle \neq 0 \text{ iff } |\psi\rangle \in \mathcal{C}$$

$$\left(\frac{1 \pm g_k}{2}\right) \left(\frac{1 \pm g_k}{2}\right) = \frac{1 \pm g_k}{2}, \quad \left(\frac{1 \pm g_k}{2}\right) \left(\frac{1 \mp g_k}{2}\right) = 0,$$

$$\mathcal{P}_{\mathcal{S}, \vec{x}}^2 = \mathcal{P}_{\mathcal{S}, \vec{x}}, \quad \mathcal{P}_{\mathcal{S}, \vec{x}_i} \mathcal{P}_{\mathcal{S}, \vec{x}_j} = \delta_{ij} \mathcal{P}_{\mathcal{S}, \vec{x}_i},$$

$$\exists \text{ unitary } g \text{ s.t. } g \mathcal{P}_{\mathcal{S}, \vec{x}_i} g^\dagger = \mathcal{P}_{\mathcal{S}, \vec{x}_j} \implies \dim [\mathcal{P}_{\mathcal{S}, \vec{x}_i}] = \dim [\mathcal{P}_{\mathcal{S}, \vec{x}_j}] \quad \forall i, j$$

$$\sum_{i=0}^{2^m-1} \mathcal{P}_{\mathcal{S}, \vec{x}_i} = \hat{\mathbb{I}} \text{ where } \vec{x}_i = \vec{x}_j \text{ iff } i = j,$$

$$\implies 2^n = \dim [\hat{\mathbb{I}}] = \sum_{i=0}^{2^m-1} \dim [\mathcal{P}_{\mathcal{S}, \vec{x}_j}] = 2^m \dim [\mathcal{P}_{\mathcal{S}, \vec{x}_0}],$$

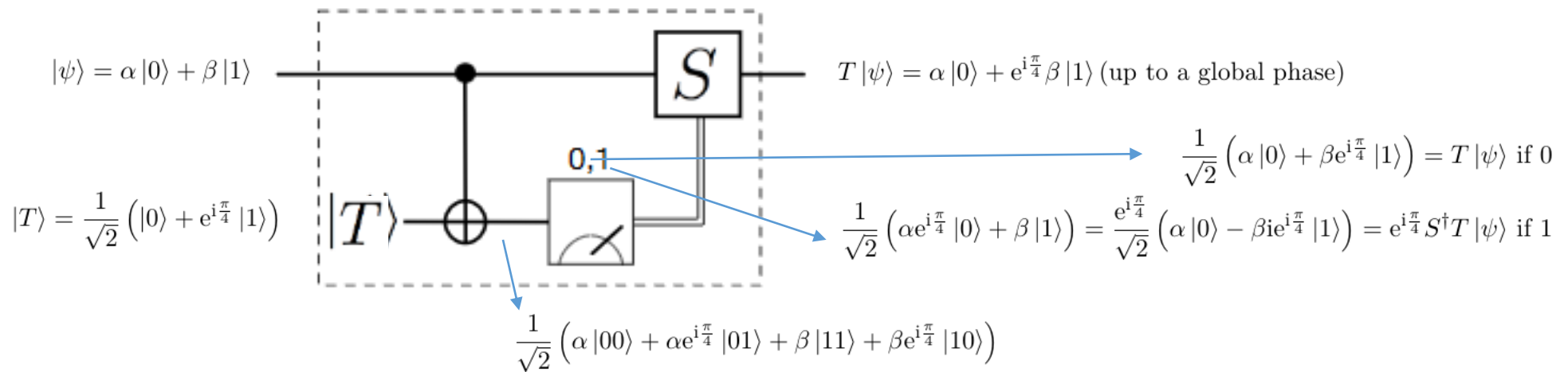
$$\implies \dim [\mathcal{C}] = \dim [\mathcal{P}_{\mathcal{S}, \vec{x}_0}] = 2^{n-m}.$$

T States

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}$$

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad \boxed{T} \quad T|\psi\rangle = \alpha|0\rangle + e^{i\frac{\pi}{4}}\beta|1\rangle$$

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$$



$$|\psi\rangle |T\rangle = \frac{1}{\sqrt{2}}(\alpha|00\rangle + \alpha e^{i\frac{\pi}{4}}|01\rangle + \beta|10\rangle + \beta e^{i\frac{\pi}{4}}|11\rangle)$$