An introduction to the surface code

A. N. Cleland[1*]

1 University of Chicago, Chicago IL 60637, USA
* anc@uchicago.edu

March 22, 2022

Abstract

This chapter provides an introduction to the surface code, discussing details about how physical qubits are connected and operated to create logical qubits, how errors are handled, and what performance metrics are needed to generate logical qubits with performance that exceeds their underlying physical qubit performance.

Contents

1 Abstract

These lecture notes provide a thorough introduction to the surface code, an architecture for building a quantum computer. The surface codes originated in the toric codes, invented by Alexei Kitaev [1–4], which in turn evolved from Kitaev's models for supporting topological order, using qubits distributed on the surface of a toroid, a two-dimensional surface with periodic boundary conditions along both in-plane directions. Kitaev's toroidal geometry ultimately turned out to be unnecessary, and the planar versions, known as the surface codes, were primarily developed by Bravyi and Kitaev as well as Freedman and Meyer [5,6].

2 Overview

The surface codes, of which there are a number of variants, provide an appealing architecture for building a quantum computer [5, 7]. This is primarily because the surface codes provide a simple architecture that promises, under certain assumptions, an exponential reduction in the error rate for logical qubits as one scales up the number of physical qubits that define each logical qubit. The surface codes originated in the toric codes, invented by Alexei Kitaev [1–4], which in turn evolved from Kitaev's models for supporting topological order, using qubits distributed on the surface of a toroid, a two-dimensional surface with periodic boundary conditions along both in-plane directions. Kitaev's toroidal geometry ultimately turned out to be unnecessary, and the planar versions, known as the

surface codes, were primarily developed by Bravyi and Kitaev as well as Freedman and Meyer [5, 6].

Error-protection schemes such as the surface codes rely on assemblies of physical qubits that are called **logical qubits**. The physical qubits that make up a logical qubit are operated in a cyclically-repeated quantum circuit that detects errors in the physical qubits as they occur. If the quantum gates in this circuit are sufficiently accurate, then the logical qubit will have a lower error rate than its underlying physical qubits. The error-protecting architecture has a certain threshold for the physical qubit error rate; for error rates above this threshold, the error-protection scheme fails (spectacularly). In general there is an assumption that *local* errors dominate, i.e. errors occurring in the individual physical qubits, uncorrelated with those occurring in other qubits; and errors in the CNOT entangling gates between pairs of physical qubits. Long-distance correlated errors, between pairs or larger groups of qubits, are in general assumed not to occur; numerical tests indicate that to some extent, such correlated errors are not disastrous, but the full panoply of possible long-range correlated errors has not been explored, and certainly some types of errors can cause propagation and failure of the surface code protection.

Note that in general errors are *not* corrected as they occur and are detected in the surface code cycle; instead, the overall control software accounts for these errors in measurement outcomes, propagating errors through the single qubit and two-qubit gates where possible.

Surface codes are unusually tolerant to local physical qubit errors. In one early estimate [8], the authors found that one type of surface code could handle error rates of almost 3% per cycle of the surface code circuit, although the authors assumed certain capabilities in the qubits that turn out to be difficult to implement in the currently available qubit technologies. Following this promising result, Raussendorf and co-workers discovered further that a logical CNOT operation, between logical qubits, could be implemented in an error-protected fashion by braiding transformations on a single surface, significantly simplifying the architecture for the surface code [9–11]. These authors also arrived at an error threshold of 0.75% per physical qubit operation, assuming only simple nearest-neighbor one- and two-qubit gates, lower than the threshold of Preskill et al., but with significantly simpler hardware requirements.

The tolerance of surface codes to errors, withstanding error rates of order 1% [12, 13], is far higher than that of other error-protecting approaches. The Steane and Bacon-Shor codes implemented on two-dimensional lattices with nearest-neighbor coupling, for example, have thresholds of about 2×10^{-5} [14, 15], three orders of magnitude lower than the surface code (i.e. the qubits need to perform three orders of magnitude better to enable implementation using e.g. a Steane code). As a result, there has been significant interest in developing further the surface codes [12, 13, 16, 17]. A number of authors are also working on improving the classical processing associated with the surface code [18–22], where error identification forms a significant software challenge. There are also a number of other two-dimensional topological codes that have appealing error thresholds and scaling, although these all tend to require physical qubits with additional capabilities [23–26].

Error protection always seems to involve large numbers of physical qubits [14, 15]; the surface code is no exception. It takes a minimum of thirteen physical qubits to implement a single surface code logical qubit (even with zero physical qubit error rate). As the local error rate increases, the number of physical qubits needed to achieve a certain logical qubit error rate increases as well. Above an error rate threshold of around 1%, the surface code no longer provides protection; however, this threshold depends on the particular error source: For example, errors in the projective measurement of physical qubits can approach 50% without ruining the surface code protection, while errors in the physical

qubit controlled-NOT must be kept below 1%. Given current physical qubit performance, it is likely that a reasonably fault-tolerant logical qubit will require of order 10^3 to 10^4 physical qubits; however, each factor reduction in the error rate for the critical CNOT gate would reduce the number of required physical qubits by about the same factor (up to a limit). The number of physical qubits needed to factor a "interesting" number using Shor's algorithm [27] depends on a tradeoff between the quantum circuit's physical size and the amount of time considered acceptable for the computation: Larger circuits take less time, but are more expensive to build and operate. Estimates for surface-code circuit sizes are provided in Ref. [28], depending on the size of the number to be factored. For a 2,000 bit number (about 600 decimal digits), if we are willing to wait about one day for the result, the computational part of Shor's algorithm will need about 4,000 logical qubits; if we assume about 3,500 physical qubits per logical qubit, this corresponds to about 15 million physical qubits. However, Shor's algorithm has to perform modular exponentiation, which requires large numbers of what are known as $|A\rangle$ states, with $|A\rangle_L = |0\rangle_L + e^{i\pi/4}|1\rangle_L$ (the L subscript indicates these are logical qubit states). These states must be prepared very precisely, with a precision given by the overall surface code requirement. For a 2,000 bit number, this means precision of order 1 part in 10^{16} or 10^{17}. This can be achieved using a method called "magic state distillation" [29–31]. Producing these high-precision states at a sufficiently high rate and purity consumes the majority of a quantum computer's resources; here roughly 10 times the number of logical qubits, and thus 10 times the number of physical qubits, are required to produce these states at a rate sufficient to complete the computation in one day. Thus of order 2×10^8 physical qubits are needed to factor a 2,000 bit number, accounting for the need for $|A\rangle$ states.

Note that using the most powerful classical supercomputer and the best classical factoring algorithm, factoring such a number would require more time than the age of the universe and more electrical power than is available on the Earth. You can try: a company called RSA Laboratories (now called RSA Security LLC) published a 617 decimal digit number that is the product of two primes, and offered 200,000 USD to whoever could factor it. That number is called RSA-617 or RSA-2048 (it has 2048 binary digits), and is equal to

25195908475657893494027183240048398571429282126204032027777137836043662020707595556264018525880784406918290641249515082189298559149176184502808489120072844992687392807287776735971418347270261896375014971824691165077613379859095700097330459748808428401797429100642458691817195118746121515172654632282216869987549182422433637259085141865462043576798423387184774447920739934236584823824281198163815010674810451660377306056201619676256133844143603833904414952634432190114657544454178424020924616515723350778707749817125772467962926386356373289912154831438167899885040445364023527381951378636564391212010397122822120720357.

Give it a try!

2.1 Experimental progress in implementing a surface code

Note: this section uses terminology not introduced until later in this chapter. For those not familiar with the surface code, the author recommends skimming this section on a first read, then returning after reading the rest of the chapter.

There are rapidly developing experimental efforts to build circuits that can test implementations of the surface code, with the most advanced efforts using superconducting qubit circuits. Evaluation of the performance of transmon-style qubits [32] in small test circuits that comprise sections of a surface code [33, 34] showed that the physical qubit

gate fidelities were very close to the break-even point for a surface code, perhaps slightly below. Note however, as discussed elsewhere, the error rate threshold is not a single number; instead the overall surface code performance has a different sensitivity to errors in each elementary physical qubit operation, with the highest sensitivity to the two-qubit entangling gate (e.g. a CNOT or CZ gate), which unfortunately is also the class of gates that has proven the most challenging to achieve the best fidelities. The size of a surface code logical qubit, i.e. the number of physical qubits required to implement a logical qubit, depends strongly on how close the physical qubit performance is to the surface code error threshold, with logical qubits growing rapidly in size as physical error rates approach the threshold from below. Keeping the circuit sizes manageable will probably require error rates, loosely speaking, one or two orders of magnitude below than the threshold, especially for the CNOT-class gates.

Nonetheless there have been very interesting and increasingly sophisticated efforts to simulate the surface code using photonic or other qubits [35–39], which we do not discuss further here. There are also efforts to build small, fully-operational surface code circuits [40–46].

First steps towards an operational surface code using superconducting qubits include early experiments using linear chains of qubits to demonstrate joint measurement and bit-flip error detection [33, 47], with first five and then nine qubits with nearest-neighbor coupling. More sophisticated later steps include building and measuring a superconducting qubit circuit with seven qubits that approximate two plaquettes of the surface code, with which measurements showed preliminary indications of the anyonic behavior of excitations in these circuits [40]. More recently, two separate experiments [41, 44] explored measurements in circuits of the same size, with four data and three measure qubits, demonstrating error detection of a logical qubit based on the four physical qubits. In one experiment [41], the logical qubit had a post-selected lifetime about four times that of the best physical qubit in the circuit. The logical qubit could be probabilistically placed in the $|0000\rangle + |1111\rangle$ state, with a measured success probability of 25 percent (using a technique that, if error-free, would yield this state 50 percent of the time). In the second experiment [44], the authors demonstrated the high fidelity preparation and measurement of the four data qubits in their four cardinal logical states $|0\rangle_L = |0000\rangle$, $|1\rangle_L = |1111\rangle$, $|+\rangle_L$ and $|-\rangle_L$.

More recently two separate accounts of more complete surface code experiments have appeared, using a seventeen qubit circuit with nine data qubits and eight measure qubits, the latter providing four X and four Z stabilizers and creating an implementation of a distance $d = 3$ surface code circuit [45, 46]. In one experiment [45], the researchers use an error correction scheme that preserves the four primary states of the logical qubit, comprising the nine data qubits. Repeated cycles allow detection of bit- and phase-flip error syndromes, with errors corrected in post-processing. When rejecting runs in which state leakage was detected, they find an error probability of about 3 percent per cycle. A very similar experiment on the same size circuit with the same surface code topology is described in [46]. By executing several consecutive error correction cycles, the experimenters found that the logical error could be significantly reduced from the native error after applying corrections by post-selecting error-free states, yielding an error-corrected logical qubit with an improved fidelity over e.g. the best single physical data qubit. Both of these experiments had mean single-qubit gate fidelities of order 99.9 percent and two-qubit gate fidelities of order 99 percent, with measurement fidelities above 90 percent, roughly speaking above the large-circuit limit for the surface code error threshold.

There has also been work on larger circuits implementing the surface code. The Google AI team has published a number of papers in this regard, the most relevant being Refs. [42]

and [43], using a circuit with 54 total superconducting qubits and conducting experiments using a subset of these. In [42], researchers implement one-dimensional repetition codes for a line of qubits embedded in a two-dimensional circuit, and demonstrate the exponential suppression of either bit- or phase-flip errors, reducing logical error rates by more than two orders of magnitude per measurement cycle when increasing the number of qubits from five to 21, a result that is stable over 50 rounds of error correction. In [43], in a second advance using a similar circuit, researchers prepared the ground state of the surface code Hamiltonian, and measured a topological entanglement entropy near the expected value of $-\ln 2$, and further simulated anyon interferometry to explore the excitation braiding statistics. They also explored key aspects of the surface code, including logical state injection and the decay of the non-local order parameter.

These various results, which by no means include all experiments exploring the surface code implemented using superconducting qubits, not to mention other qubit hardware platforms, point to steadily advancing capabilities in experimental implementations of the surface code. As qubit energy and phase lifetimes improve, and correspondingly the single and two-qubit fidelities improve, the fundamental error rates will likely continue to improve beyond the surface code threshold, and researchers will then have to begin to deal with long-range errors, either due to circuit design or due to external influences [48–50].

3 Requirements for a quantum computer

A quantum computer requires a set of inter-connected physical qubits (quantum two-level systems, with basis states $|0\rangle$ and $|1\rangle$), and the ability to perform unitary operations (gates) on these qubits, as well as measure their quantum states with a projective measurement (one where after the measurement, the qubit state is an eigenstate of the measurement operator). The interconnectedness of the qubits is often not discussed, and many algorithms assume that the physical qubits have all-to-all coupling (every qubit connected to every other), which requires $N(N-1)/2$ interconnects for N qubits. Architectures with less than all-to-all coupling require additional computational overhead. The surface code, as we will see, is a planar square array requiring only nearest-neighbor *physical* connections. However, as we will see, the surface code architecture places no constraints on the *logical* qubit interconnections: All-to-all coupling is easily achieved in the surface code, greatly simplifying the algorithm design.

We now begin with an introduction to quantum computing, sufficient to understand the surface code. We will write the physical qubit states as $|0\rangle$ and $|1\rangle$. These are the eigenstates of the Pauli $\hat{Z}$ operator with eigenvalues $+1$ and -1 respectively, are usually chosen to be the basis states for the vector space. These are thus assigned the column vectors

$$|0\rangle \leftrightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle \leftrightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{1}$$

Using this basis, the most important single qubit operators (called "gates" in quantum computing) are most easily represented by matrices, given here in the Z basis for completeness:

$$\text{Identity or } \hat{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \tag{2}$$

$$\hat{\sigma}_X \text{ or } \hat{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \tag{3}$$

$$\hat{\sigma_Y} \text{ or } \hat{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \tag{4}$$

$$\hat{\sigma_Z} \text{ or } \hat{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{5}$$

(you can check that the column vectors for $|0\rangle$ and $|1\rangle$ are indeed the eigenvectors of the $\hat{Z}$ operator with the correct eigenvalues). Note that the eigenstates of the $\hat{X}$ operator are

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \leftrightarrow \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \tag{6}$$

and

$$|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \leftrightarrow \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \tag{7}$$

with eigenvalues $+1$ and -1 respectively. Note we also have

$$|0\rangle = \frac{1}{\sqrt{2}}(|+\rangle + |-\rangle) \tag{8}$$

and

$$|1\rangle = \frac{1}{\sqrt{2}}(|+\rangle - |-\rangle). \tag{9}$$

Note also that in some quantum computing texts the imaginary i is dropped in the $\hat{Y}$ operator, which can cause some confusion.

The Hadamard operator is

$$\hat{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{10}$$

not to be confused with the Hamiltonian. The $\hat{S}$ gate (also called the $\pi/4$ gate) is given by

$$\hat{S} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} = e^{i\pi/4} \begin{pmatrix} e^{-i\pi/4} & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}, \tag{11}$$

and the $\hat{T}$ gate (also called the $\pi/8$ gate) is

$$\hat{T} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix} = e^{i\pi/8} \begin{pmatrix} e^{-i\pi/8} & 0 \\ 0 & e^{i\pi/8} \end{pmatrix}. \tag{12}$$

Problem 1: Find the eigenvectors and eigenvalues for the $\hat{H}$, $\hat{S}$ and $\hat{T}$ operators.

We will be dealing with multi-qubit states. A two-qubit product state for qubits a and b, where qubit a is in the state $|u\rangle$ and qubit b is in the state $|v\rangle$, can be written $|u_a\rangle \otimes |v_b\rangle = |u_a v_b\rangle$ (here $\otimes$ means outer product). This is a state vector living in a $2 \times 2 = 4$ dimensional vector space. We will use the shorthand $|uv\rangle$ to represent this state, where the order of the letters corresponds to the order of the qubits; hence $|vu\rangle$ would represent the state $|v_a\rangle \otimes |u_b\rangle$. Most two-qubit states are not product states; for example $(|uv\rangle + |vu\rangle)/\sqrt{2}$ is an entangled state that cannot be written as the product of two one-qubit states. The existence of these kinds of states was the subject of a famous and prolonged discussion between Neils Bohr and Einstein, and gave rise to the latter's involvement in the writing of the "EPR paradox" paper [51] as well as a series of letters between Einstein and Schrödinger (for an interesting discussion, see [52]).

We can now discuss the controlled-not (CNOT) gate, which operates on two qubits. For a CNOT, the first qubit is called the control qubit, and performs a conditional $\hat{X}$ gate on the second qubit, which is called the target. The $\hat{X}$ gate is performed if the control qubit is in its $|1\rangle$ state, otherwise nothing (an $\hat{I}$ gate) is done to the target qubit. As an $\hat{X}$ gate can be seen as a NOT gate, flipping the state of its target qubit, the controlled application of this gate provides the name for the CNOT gate.

Using the canonical two-qubit basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, the matrix representation of the CNOT gate is

$$CNOT \leftrightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{13}$$

You can check that the CNOT gates performs the controlled-X operation $|10\rangle \rightarrow |11\rangle$ and $|11\rangle \rightarrow |10\rangle$, leaving the other states unchanged.

Problem 2: Find the eigenvectors and eigenvalues of the CNOT operator.

Which set of gates one actually needs depends on what is most easily implemented with the physical qubits. The Solovay-Kitaev theorem [2, 53] implies that a complete set of gates to build a quantum computer comprises the single-qubit operators $\hat{I}$, $\hat{X}$, $\hat{Z}$, the Hadamard $\hat{H}$, the $\hat{S}$ and $\hat{S}^\dagger$ phase gates, and the $\hat{T}$ and $\hat{T}^\dagger$ gates. A two-qubit entangling gate is also needed, which is usually chosen to be the CNOT gate (any two-qubit gate from which a CNOT can be constructed is of course acceptable). Note that as we have the identities $\hat{T}^2 = \hat{S}$, $\hat{T}^4 = \hat{Z}$, $\hat{H}\hat{Z}\hat{H} = \hat{X}$, $\hat{Z}\hat{S} = \hat{S}^\dagger$, and $\hat{T}^7 = \hat{T}^\dagger$, the *minimal* set of gates is $\{T, H, CNOT\}$.

Note: From hereon in, we will drop the $\hat{X}$ notation in favor of the simpler X notation, where the context will hopefully always be clear.

In the surface code, we must be able to swap the states of two qubits, an operation that is imaginatively called a SWAP operation. A SWAP operation, taking the two qubit state $|uv\rangle$ to $|vu\rangle$, can be performed with three CNOT operations. This is shown (and discussed) later; see Fig. 11.

All of the gates we have discussed so far are unitary. In case your quantum mechanics is rusty, a unitary operator U is one where U times its adjoint $U^\dagger$ is the identity operator I, i.e.

$$UU^\dagger = U^\dagger U = I. \tag{14}$$

To form the adjoint of an operator, you take the complex conjugate transpose of the operator's matrix representation. As you can see, except for the S and T gates, the operators we have introduced so far are all their own adjoints, and their squares are the identity matrix. These are thus by definition Hermitian.

Products of operators appear frequently. The order of a product matters, as in general $AB \neq BA$ when dealing with operators (i.e. matrices). If $AB = BA$, then A and B are said to commute:

$$[A, B] = AB - BA = 0, \tag{15}$$

where $[A, B]$ is the **commutator** of A and B.

Qubit operators in general do not commute. The identity I commutes with all operators. However, you can easily show by multiplying the matrices for X and Z that

$$XZ = -ZX, \tag{16}$$

so that the commutator

$$[X, Z] = XZ - ZX = 2XZ = -2ZX, \tag{17}$$

as you can verify by multiplying the matrices out. This means that operating on a qubit first with Z then with X gives the negative of operating first with X and then with Z.

However, a very curious commutation feature occurs with two-qubit states. If we have qubit a and b, with their respective X and Z operators X_a, X_b, Z_a and Z_b, as we have seen, X_a and Z_a do not commute, and similarly for X_b and Z_b. However, operators on different qubits always commute, so X_a and Z_b for example commute. A surprising fact is that the *product* of the operators X_aX_b commutes with the product of operators Z_aZ_b, as you can show:

$$\begin{aligned} X_aX_bZ_aZ_b &= X_aZ_aX_bZ_b = (-Z_aX_a)(-Z_bX_b) \\ &= Z_aX_aZ_bX_b = Z_aZ_bX_aX_b, \end{aligned} \tag{18}$$

using the fact that operators on different qubits always commute. Hence

$$[X_aX_b, Z_aZ_b] = 0. \tag{19}$$

Note that when we write ZX without subscripts, the Z and X operators act on the same qubit, so ZX is represented by a 2×2 matrix resulting from the matrix product of the two 2×2 matrices representing Z and X. Thus

$$ZX \leftrightarrow \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}. \tag{20}$$

By contrast, when we write Z_aX_b, which more properly we should write as $Z_a \otimes X_b$, these operators act on different qubits, so the representation of this *outer product* operator is a 4×4 matrix acting on a two-qubit vector space. The "canonical" basis for this two-qubit space is the same as what we used for the CNOT operator, namely $\{|0_a0_b\rangle, |0_a1_b\rangle, |1_a0_b\rangle, |1_a1_b\rangle\}$. The representation of the outer product operator Z_aX_b in this basis is

$$Z_aX_b \leftrightarrow \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{21}$$

Problem 3: Find the eigenvectors and eigenvalues of Z_aX_b in the canonical basis.
Problem 4: Write the matrix representation of X_aZ_b in the canonical basis.

3.1 Measurement

We haven't talked about measurement. In the surface code, all measurements are *projective*; you should recall from your quantum mechanics classes that this means that a measurement of an operator projects the qubit onto an eigenstate of the measurement operator, and the measurement value that is returned is the corresponding operator eigenvalue. For the surface code, you only need to be able to perform single-qubit Z measurements, projecting the qubit onto the $|0\rangle$ or $|1\rangle$ state with a measurement result of $+1$ and -1, respectively. It is useful (but not necessary) to also be able to perform single-qubit X measurements, returning $+1$ and leaving the qubit in the $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$

state, or returning -1 and leaving the qubit in the $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$ state. In fact, however, an X measurement capability is not needed because the Hadamard H operator very conveniently maps $|+\rangle \leftrightarrow |0\rangle$ and $|-\rangle \leftrightarrow |1\rangle$ (a second applications of H gives the identity). Hence, if an X measurement is needed, you simply apply H to the qubit, do a Z measurement, and perform H on the state resulting from the measurement. The final state is then an X eigenstate and the measurement value corresponds to that eigenstate.

If two operators commute, then quantum mechanics guarantees that there is a basis that comprises eigenvectors for both operators. Consider our two-qubit commuting operators X_aX_b and Z_aZ_b. The first pair has an eigenstate $|++\rangle$ and the second an eigenstate $|11\rangle$ (dropping the subscripts a and b and relying on order only), but these are not eigenstates for the other pair. However, as you can easily show, the Bell states $\{|00\rangle + |11\rangle, |01\rangle + |10\rangle, |01\rangle - |10\rangle, |00\rangle - |11\rangle\}$ (divide by $\sqrt{2}$ to normalize) are all eigenstates of both operators, with eigenvalues $\{+1, -1, -1, +1\}$ for Z_aZ_b as can easily be seen, and eigenvalues $\{+1, +1, -1, -1\}$ for X_aX_b. The eigenvalues for X_aX_b are most easily seen by re-writing the eigenstates as $\{|00\rangle + |11\rangle = |++\rangle + |--\rangle, |01\rangle + |10\rangle = |++\rangle - |--\rangle, |01\rangle - |10\rangle = |+-\rangle - |-+\rangle, |00\rangle - |11\rangle = |-+\rangle + |+-\rangle\}$ (all divided by $\sqrt{2}$). Note that the two eigenvalues (for Z_aZ_b and X_aX_b) identify the precise Bell state.

Problem 5: Show that the Bell states can be written as claimed using the $\{|+\rangle, |-\rangle\}$ basis vectors.

4 Quantum Circuits

A *quantum circuit* comprises a sequence of gates that operate on a population of qubits. They are the method by which a quantum algorithm is implemented, and provide a kind of "machine language" for a quantum computer. They are usually represented graphically. Figure 1 displays the operation $|u'\rangle = H|u\rangle$. The qubit evolution is represented by a horizontal time-line, where time moves forward from left (starting with $|u\rangle$) to the right through one or more boxes (here, the Hadamard H) that represent the gates applied to each qubit, ending in the final qubit state ($|u'\rangle$). A one-qubit gate has one qubit time-line entering and one emerging; a two-qubit gate has two such lines.

Figure 1: Quantum circuit for a single qubit, where a single Hadamard gate transforms $|u\rangle$ to $|u'\rangle = H|u\rangle$.

In Fig. 2 we add a second gate after the Hadamard, representing the operation $|u'\rangle = XH|u\rangle$. Note that as H is applied first to the qubit, it appears to the left of (earlier than) the second operation X.

Figure 2: Quantum circuit for $|u'\rangle = XH|u\rangle$.

For multiple qubits you run lines in parallel, one for each qubit. The circuit in Fig. 3 has two qubits, giving for the upper qubit time-line $|u_1'\rangle = XH|u_1\rangle$ and for the lower

$|u_2'\rangle = HX|u_2\rangle$.

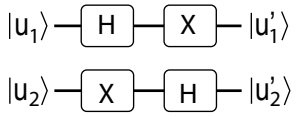

Figure 3: Quantum circuit for two qubits in parallel.

Next we look at a two-qubit CNOT, which is a single box interrupting the timelines of both qubits. The quantum circuit graphical representation of a CNOT is shown in Fig. 4. This circuit represents the operation $|u'\rangle = CNOT(|u_1\rangle, |u_2\rangle)$, where $|u'\rangle$ is the two-qubit (bipartite) output state, and $|u_1\rangle$ the state of the control qubit while $|u_2\rangle$ the state of the target qubit.

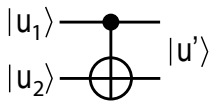

Figure 4: Quantum circuit for a CNOT; the smaller solid circle indicates the control qubit, and the larger cross-circle the target qubit.

The operator representation for the CNOT gate, as mentioned earlier, can be written

$$CNOT = |0\rangle\langle 0| \otimes I_2 + |1\rangle\langle 1| \otimes X_2, \tag{22}$$

where I_2 and X_2 operate on the second, target qubit.

It is often useful to combine multiple CNOT gates, involving one control qubit and multiple targets, into a single operation as shown in Fig. 5. This is called a single-control, multi-target CNOT. Do not confuse this circuit with the *Toffoli gate*, which has two controls and a single target, which we show in Fig. 6. This is a straightforward extension of the CNOT gate, with two control qubits and one target qubit.

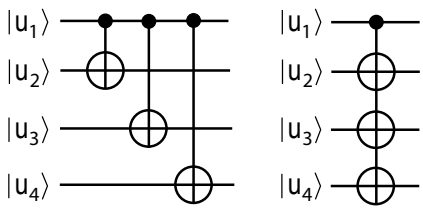

Figure 5: A multi-target, single control circuit (left) can be compacted into the circuit on the right.

The operator representation for the Toffoli is

$$cU = (I_1 \otimes I_2 - |11\rangle\langle 11|) \otimes I_3 + |11\rangle\langle 11| \otimes X_3, \tag{23}$$

where I_j is the identity for the jth qubit and X_3 the X operator for the target qubit.

The Toffoli can be constructed from single qubit gates and a few CNOTs; the minimal circuit to implement the Toffoli involves six CNOTs and a number of Hadamard and T

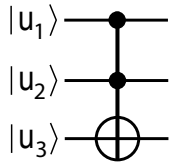

Figure 6: Quantum circuit for a Toffoli gate; the first two qubits $|u_1\rangle$ and $|u_2\rangle$ are the combined controls, and the third qubit $|u_3\rangle$ the target.

and $T^\dagger$ gates, shown in Fig. 7. Hence in principle the Toffoli gate itself is not needed, i.e. it is not a "fundamental" gate; however, it is very useful, and "expensive" to implement with more fundamental gates, so a hardware system that supports the Toffoli can afford some computational efficiency.

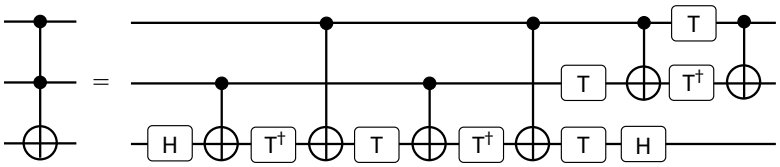

Figure 7: Toffoli gate implemented with single- and two-qubit gates.

A general two-qubit unitary gate is drawn as shown in Fig. 8, where the circuit is drawn assuming the two qubits entering the gate are in a product state, although this is not necessary, while the output state $|u'\rangle$ would in general be a two-qubit entangled state.

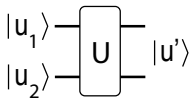

Figure 8: Two-qubit unitary gate.

Next in Fig. 9 we show a controlled-U (cU) gate, where the unitary U operation is performed when the control qubit is in the $|1\rangle$ state, but if the control is in $|0\rangle$, the identity I is applied.

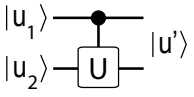

Figure 9: Controlled-U circuit.

The operator representation for this gate is

$$cU = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U. \tag{24}$$

We're now ready to do some examples. Consider the circuit shown in Fig. 10. Both qubits start in some state $|xy\rangle$ (where x and y are each either 0 or 1), a Hadamard is

applied to the first qubit, and then a CNOT is performed with the first qubit as control and the second as target. What does this circuit do? Obviously from the caption it generates Bell states; how does this work?

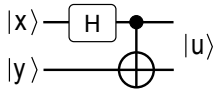

Figure 10: Bell-state generating circuit.

First consider what the circuit does for two possible choices of x and y, with the arrows indicating the outcome after each gate (first after applying the Hadamard, then after applying the CNOT):

$$|00\rangle \Rightarrow (|00\rangle + |10\rangle)/\sqrt{2} \Rightarrow (|00\rangle + |11\rangle)/\sqrt{2} = |\beta_{00}\rangle, \tag{25}$$

$$|01\rangle \Rightarrow (|01\rangle + |11\rangle)/\sqrt{2} \Rightarrow (|01\rangle + |10\rangle)/\sqrt{2} = |\beta_{01}\rangle. \tag{26}$$

Can you figure out what the outcomes are for the other values of x and y? A certain pattern should emerge.

Another way to analyze this circuit is using operator representations, combining the Hadamard on the first qubit with nothing on the second ($U_{\rm HI} = H \otimes I$), followed by a CNOT on the second qubit, $U_{\rm CNOT} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X$:

$$\begin{aligned} U_{\rm CNOT}U_{\rm HI} &= (|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X)\,(H \otimes I) \\ &= |0\rangle\langle +| \otimes I + |1\rangle\langle -| \otimes X. \end{aligned} \tag{27}$$

You can check that gives the correct results for different input states. Note to get this result, we operated with the Hadamard from the *right* on the bras on the *left*. What the notation means is that if you apply this operator to the input state $|xy\rangle$, the first part of the operator projects out the $|+\rangle$ component of $|x\rangle$, and that component is output as $|0y\rangle$, while the second part projects out the $|-\rangle$ component of $|x\rangle$ and outputs $|1, Xy\rangle$, using this shorthand to indicate applying an X operator to the $|y\rangle$ state.

Finally, we look at the circuit that implements a SWAP gate, a necessary operation for the surface code, using only two-qubit CNOT gates. This is shown in Fig. 11. This will take a product state $|uv\rangle$ and generate the state $|vu\rangle$, and works in general, i.e. will also successfully swap the qubits in an entangled state.

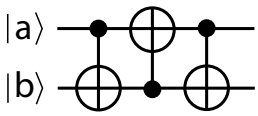

Figure 11: Two-qubit swap circuit.

Problem 6: Show that the circuit in Fig. 11 indeed performs a SWAP operation as claimed.

4.1 Measurement

Measurement is a process we have discussed earlier. We previously stated that, for the surface code, a measurement of a particle's (or qubit's) quantum state is projective, and

depending on the measurement result the qubit ends up in one of two corresponding eigenstates. As a result, the qubit no longer has any useful (unknown) information in it, and from the point of view of a quantum algorithm, the qubit has effectively been destroyed. While the physical qubit of course can be re-used, the quantum circuit representation of a qubit measurement is a qubit time-line entering from the left, which then *ends* in the circuit element that represents measurement. This is shown in Fig. 12, where the classical information that emerges from the measurement is represented by two light-weight lines, in contrast to the single heavy line used for the qubit.

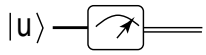

Figure 12: Measurement circuit. A quantum state $|u\rangle$ (single line) enters from left, and classical information (two lines) emerges from the right. Note this process is irreversible, unlike almost all the other quantum gates.

5 Errors and a qubit named Angelina

If all the gates and measurements could be performed exactly, without any errors, we wouldn't need the surface code. However, qubits are subject to many things that can cause errors. The most trivial are "clumsiness" errors, where you have mis-calibrated your equipment and for example your X gate includes some admixture of another gate, maybe a tiny piece of a Z gate, so maybe the gate is actually $X + cZ$ where c is small but not known, generating a superposition output state, $|0\rangle \to |1\rangle + c|0\rangle$ (note we're being sloppy with normalization).

More pernicious errors are unwanted interaction errors. When we listed the gates, we explicitly included the identity gate I; this is a very important gate, because whenever you are not applying some intentional gate to a qubit, you are implicitly acting on the qubit with the I gate. It is very hard to do the I gate precisely, even if this involves doing nothing to the qubit: Just as "idle hands are the devil's workshop" [54], so idle qubits are the targets of all kinds of bad deeds. Maybe the idle qubit interacts with its neighboring qubits, because of stray coupling; maybe it absorbs a photon wandering in from the environment; maybe it emits a photon into the environment. All of these non-idle processes will generate entanglement, with another qubit, or with the environment. These errors can cause serious problems with a computation.

To see how to handle errors, we first review the density matrix representation of a qubit state. For this, let's introduce a qubit we'll call Angelina. In the simplest case, Angelina is in a pure superposition state,

$$|A\rangle = a|0\rangle + b|1\rangle, \tag{28}$$

where a and b are complex amplitudes whose square magnitudes reflect the probability of finding Angelina in the $|0\rangle$ or $|1\rangle$ states, with

$$p(0) = |a|^2, \quad p(1) = |b|^2. \tag{29}$$

If we prefer the Bloch sphere representation, we can write $a = |a|e^{i\alpha} = \cos(\theta/2)\, e^{i\alpha}$ and $b = |b|e^{i(\alpha+\phi)} = \sin(\theta/2)e^{i(\alpha+\phi)}$, so that Angelina's state, dropping the overall unmeasurable angle α, is

$$|A\rangle = \cos(\theta/2)|0\rangle + \sin(\theta/2)e^{i\phi}|1\rangle. \tag{30}$$

This is a vector of unit magnitude pointing somewhere on the Bloch sphere, as in Fig. 13.

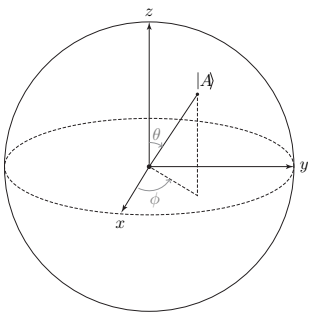

Figure 13: Quantum state $|A\rangle$ on the Bloch sphere.

This describes Angelina when she is in a pure state $|A\rangle$, i.e. when we know exactly what state $|A\rangle$ she is in. However, we may not have enough information about Angelina to know what her exact state is; instead, we may know that she is in state $|A_1\rangle$ with probability p_1, state $|A_2\rangle$ with probability p_2, etc., in other words in a *mixed state*. In this situation we represent Angelina's quantum state via the density operator ρ, given by

$$\rho = \sum_j p_j |A_j\rangle\langle A_j|, \tag{31}$$

involving the weighted sum of the kets $|A_j\rangle$ in outer product with the bras $\langle A_j|$, with weights given by the probabilities p_j. Don't confuse the states $|A_j\rangle$ with the basis states: The mixed state can involve more than the two states it takes to span the Hilbert space, and correspondingly the states do not have to be mutually orthogonal. The density matrix representation, introduced by von Neumann, is useful in part because the expectation value for measuring some operator B is simply given by

$$\langle B\rangle = tr(\rho B) = \sum_j (\rho B)_{jj}, \tag{32}$$

taking the matrix product of the representations for ρ and B and summing its diagonal elements (that's the meaning of the trace tr).

For the pure state $|A\rangle$, Angelina's density operator is simply

$$\begin{aligned}\rho = |A\rangle\langle A| = \cos^2(\theta/2)|0\rangle\langle 0| &+ \cos(\theta/2)\sin(\theta/2)e^{-i\phi}|0\rangle\langle 1| + \\ &+ \cos(\theta/2)\sin(\theta/2)e^{i\phi}|1\rangle\langle 0| + \sin^2(\theta/2)|1\rangle\langle 1|.\end{aligned} \tag{33}$$

Note this is a $2\times 2 = 4$ density matrix. This has the standard representation

$$\rho \leftrightarrow \begin{pmatrix} \cos^2(\theta/2) & \cos(\theta/2)\sin(\theta/2)e^{-i\phi} \\ \cos(\theta/2)\sin(\theta/2)e^{i\phi} & \sin^2(\theta/2) \end{pmatrix}. \tag{34}$$

For a general mixed state, the representation of the density matrix is

$$\rho \leftrightarrow \begin{pmatrix} \langle 0|\rho|0\rangle & \langle 0|\rho|1\rangle \\ \langle 1|\rho|0\rangle & \langle 1|\rho|1\rangle \end{pmatrix}. \tag{35}$$

Note that the trace of ρ is unity for both the pure and the mixed states, as you can easily show, but the trace of ρ^2 is unity only for a pure state, and is less than 1 for a mixed state.

Problem 7: Prove that a mixed state with density matrix ρ has $tr(\rho^2) < 1$.

How does Angelina's state become mixed, if we start with a pure state? Certainly this won't happen if all we do is operate on Angelina with unitary operators U, as unitary operators always take pure states to pure states.

So how does Angelina's state become mixed? One way this could happen is if we turn our backs, and our "friend" comes and, based on some coin flips, either does or does not do some unitary operations on Angelina. If the probability to perform U_j is p_j, where U_j is a unitary, then Angelina's initially pure state density operator $\rho = |A\rangle\langle A|$ will be transformed to

$$\rho' = \sum_j p_j U_j \rho U_j^\dagger = \sum_j p_j |A_j\rangle\langle A_j|, \tag{36}$$

where $|A_j\rangle = U_j|A\rangle$ and $\langle A_j| = \langle A|U_j^\dagger$. Hence, for example, say there is a 50% chance our frenemy did nothing, and a 50% chance he did an X gate. After this probabilistic manipulation, Angelina's state is given by the density matrix ρ', where

$$\rho' = \frac{1}{2}\,|A\rangle\langle A| + \frac{1}{2}\,X|A\rangle\langle A|X, \tag{37}$$

(note $X = X^\dagger$). The trace of ρ' is still unity, but the trace of ρ'^2 is 1/2 (do you see why?), so this is now a mixed state.

We of course do not have to rely on a friend to come to transform Angelina's state to a mixed one. Angelina will be interacting with her environment, which will involve many small interactions with a very complex system, very much like a classical Brownian particle interacting with its surrounding fluid. In the latter case, the Brownian particle is visibly affected by the thousands of fluid molecules that interact with it; however the accompanying particle's reaction on the fluid is not easily detected. The resulting motion, to first order, is the random walk and thermalization of the Brownian particle. Here, we similarly need to understand how Angelina interacts with the environment that surrounds her, and how she becomes entangled with that environment. How can we hope to get a handle on something we know so little about?

Let's begin by looking at something simpler. Let's introduce a second qubit, which we'll call Brad. Let's assume Angelina and Brad interact through a simple exchange Hamiltonian:

$$H_{\rm int} = \hbar g X_A X_B, \tag{38}$$

where $X_{A,B}$ are the X operators on Angelina and Brad, respectively, and g is the interaction strength in frequency units. If we start in the state $|01\rangle$, where Angelina is in $|0\rangle$ and Brad in $|1\rangle$, then an application of the Hamiltonian generates the result

$$H_{\rm int}|01\rangle = \hbar g|10\rangle, \tag{39}$$

swapping Angelina and Brad's states. Of course, if we start in the reverse situation, we get the reverse outcome. Note that the eigenstates of this Hamiltonian are the entangled Bell states $|B_{xy}\rangle$,

$$\begin{aligned}
|B_{00}\rangle &= \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right), \\
|B_{01}\rangle &= \frac{1}{\sqrt{2}}\left(|01\rangle + |10\rangle\right), \\
|B_{10}\rangle &= \frac{1}{\sqrt{2}}\left(|00\rangle - |11\rangle\right), \\
|B_{11}\rangle &= \frac{1}{\sqrt{2}}\left(|01\rangle - |10\rangle\right).
\end{aligned} \tag{40}$$

The $H_{\rm int}$ eigenvalue for the eigenstate $|B_{xy}\rangle$ is $(-1)^x\hbar g$, as was discussed earlier.

Consider now a general state $|M\rangle$ of Angelina and Brad; this evolves in time under the interaction Hamiltonian according to Schrödinger's equation,

$$i\hbar\frac{d|M(t)\rangle}{dt} = H_{\rm int}|M(t)\rangle = \hbar g X_A X_B|M(t)\rangle. \tag{41}$$

The four entangled Bell states are stationary states, with

$$|B_{xy}(t)\rangle = \exp\left((-1)^x igt\right)\,|B_{xy}(0)\rangle. \tag{42}$$

If we start say with the state $|10\rangle = (|B_{01}\rangle - B_{11}\rangle)/\sqrt{2}$ at time $t=0$, the state will become entangled, with a time dependence

$$|M(t)\rangle = \frac{1}{\sqrt{2}}\left(e^{igt}|B_{01}\rangle - e^{-igt}|B_{11}\rangle\right) \tag{43}$$

$$= \frac{e^{igt}}{\sqrt{2}}\left(|B_{01}\rangle - e^{-2igt}|B_{11}\rangle\right). \tag{44}$$

Problem 8: Show that this is indeed the solution to Schrödinger's equation, Eq. 41.

The state transforms periodically to the state $|01\rangle = (|B_{01}\rangle + B_{11}\rangle)/\sqrt{2}$ (with a phase factor), every $t_n = (n+1/2)\pi/g$, and returns to the initial state $|10\rangle$ every $t'_n = n\pi/g$: The "swap time" is $T_s = \pi/g$. Other than at these precise times, however, Angelina and Brad are in an entangled state. For times Δt very short compared to the swap time, the entanglement is weak, with a state approximately given by $|10\rangle + 2ig\Delta t|01\rangle$ (needing normalization, of course), with $2g\Delta t = 2\pi\Delta t/T_s \ll 1$. From the point of view of Angelina, this is a small X_A error; measuring her state (ignoring that of Brad) after the time Δt will return the (erroneous) state $|0\rangle_A$ with probability $p = |2\pi\Delta t/T_s|^2 \ll 1$, and will otherwise return the original state $|1\rangle_A$ with probability $1-p \approx 1$.

Problem 9: Write the matrix that represents the Angelina-Brad interaction Hamiltonian $H_{\rm int}$ in the canonical basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. Now write the Hamiltonian in the Bell state basis. Discuss what you find.

The evolution we have seen holds for a general state, regardless of the initial states of Angelina and Brad: After a short interaction time $\Delta t \ll T_s$, the probability that Angelina's state suffers an X_A error is $p = |2\pi\Delta t/T_s|^2 \ll 1$, while the probability the original state is instead recovered on measurement is $1-p$.

Problem 10: Demonstrate that the probability of an X_A error is indeed $p = |2\pi\Delta t/T_s|^2$.

For this example, we chose the interaction between Angelina and Brad to be an XX interaction, and the interaction was deterministic: We know exactly the Hamiltonian through which they interact. We could instead have chosen for example a ZZ or YY interaction, each of which would have given a comparable result, generateing small Z_A or Y_A errors, respectively.

Problem 11: Demonstrate that a ZZ interaction will give a Z_A error, of similar form to the XX interaction.

Another assumption we made is that Brad was in a pure state when we started the interaction. Can we inject some uncertainty by instead saying that we don't know Brad's

state, that instead Brad is in a mixed state? In that case, we have to use a density matrix ρ_B to describe Brad, but it turns out this doesn't really change things in a substantial way. We can use a procedure known as *purification* to handle this situation.

Sidebar: Two-state purification

The way purification works is as follows: Say in our original basis $\{|p\rangle, |r\rangle\}$, Brad's density matrix is

$$\rho_B = \begin{pmatrix} |a|^2 & b \\ b^\star & |c|^2 \end{pmatrix}, \tag{45}$$

where $|a|^2 + |c|^2 = 1$ and the density matrix is Hermitian (note this is not meant to be the density matrix for a pure state). The Hermitian property of ρ_B means we can find another basis $\{|s\rangle, |t\rangle\}$ in which the density matrix is diagonal:

$$\rho'_B = \begin{pmatrix} |f|^2 & 0 \\ 0 & |g|^2 \end{pmatrix}, \tag{46}$$

with $|f|^2 + |g|^2 = 1$. Note the diagonal elements are real and non-negative. Written in outer product form, this is

$$\rho'_B = |f|^2|s\rangle\langle s| + |g|^2|t\rangle\langle t|. \tag{47}$$

We extend the vector space $\{|s\rangle, |t\rangle\}$ to a four-dimensional one, adding two additional basis vectors $|u\rangle$ and $|v\rangle$, such that the set $\{|s\rangle, |t\rangle, |u\rangle, |v\rangle\}$ is an orthonormal spanning set. We define the *pure* state $|B'\rangle$ for Brad in this extended space as

$$|B'\rangle = |f||s\rangle|u\rangle + |g||t\rangle|v\rangle = |f||su\rangle + |g||tv\rangle. \tag{48}$$

Note we only pair up $|s\rangle$ with $|u\rangle$ and $|t\rangle$ with $|v\rangle$, and we have taken the square roots of the 2×2 density matrix elements. The corresponding pure state density matrix is

$$\sigma_B = |B'\rangle\langle B'| = |f|^2|su\rangle\langle su| + |f||g||su\rangle\langle tv| + |g||f||tv\rangle\langle su| + |g|^2|tv\rangle\langle tv|. \tag{49}$$

If we trace out the two imaginary degrees of freedom $|u\rangle$ and $|v\rangle$, we recover the original mixed state density matrix:

$$tr_{u,v}(\sigma_B) = \langle u|\sigma|u\rangle + \langle v|\sigma|v\rangle = |f|^2|s\rangle\langle s| + |g|^2|t\rangle\langle t| = \rho'_B, \tag{50}$$

as desired.

How does Brad's extended state interact with Angelina? If we consider the $X_A X_B$ interaction, we've done two things that affect the X_B part of this: We've changed basis vectors to make Brad's density matrix diagonal, and we've extended Brad's Hilbert space from two to four dimensions. Specifically, the new basis vectors are

$$|s\rangle = U|p\rangle, \quad |t\rangle = U|r\rangle, \tag{51}$$

where U is the unitary change-of-basis operator. The components of U in the $\{|p\rangle, |r\rangle\}$ basis are

$$U \leftrightarrow \begin{pmatrix} \langle p|U|p\rangle & \langle p|U|r\rangle \\ \langle r|U|p\rangle & \langle r|U|r\rangle \end{pmatrix}. \tag{52}$$

As we have the identity

$$|p\rangle\langle p| + |r\rangle\langle r| = I, \tag{53}$$

we can write

$$\begin{aligned} |s\rangle &= IUI|p\rangle = (|p\rangle\langle p| + |r\rangle\langle r|)\, U\, (|p\rangle\langle p| + |r\rangle\langle r|)\, |p\rangle \\ &= (|p\rangle\langle p| + |r\rangle\langle r|)\, U\, |p\rangle \\ &= (\langle p|U|p\rangle)\, |p\rangle + (\langle r|U|p\rangle)\, |r\rangle, \end{aligned} \tag{54}$$

showing that the components of $|s\rangle$ in the $\{|p\rangle, |r\rangle\}$ basis are the first column of the matrix representation of U in that same basis. A similar calculation shows that the components of $|t\rangle$ are the second column of that matrix representation.

The representation of X_B in the new basis will no longer look like an X Pauli matrix, but instead will be some weighted sum of the four Pauli matrices I_B, X_B, Y_B and Z_B (now in the new basis), because these span the space of Hermitian 2×2 matrices. To deal with the extension of Brad's vector space to four dimensions, we define in the extended space the extended X'_B operator

$$X'_B = X_B \otimes I = \begin{pmatrix} X_{B,11} & X_{B,12} & 0 & 0 \\ X_{B,21} & X_{B,22} & 0 & 0 \\ 0 & 0 & X_{B,11} & X_{B,12} \\ 0 & 0 & X_{B,21} & X_{B,22} \end{pmatrix}, \tag{55}$$

where I is the identity in the extension space, and in the expression on the right, the non-zero sub-matrices are the representation of X_B in the new $\{|s\rangle, |t\rangle\}$ basis that diagonalizes the density matrix. The interaction $X_A X_B \Rightarrow X_A X'_B$, and for example the expectation value $\langle X_A X_B \rangle$ would be calculated as

$$\langle X_A X_B \rangle \Rightarrow tr_{A,B'} \left(\rho_A \sigma_B X_A X'_B \right), \tag{56}$$

tracing over Angelina's two-dimensional space as well as Brad's four-dimensional purified space. Just tracing over the extension of Brad's space will return the mixed density matrix ρ_B for Brad. In any case, the result is the same as for a pure state.

What about more generally? What if Angelina interacted with two two-level systems at the same time, say with Brad and Charlie, through a similarly deterministic Hamiltonian interaction? Or, more likely, what if she interacts with a set of harmonic oscillators, at different frequencies and with different types of interactions? Clearly these questions can go on indefinitely, building more and more complex models for an environment that in fact we know very little about. What we really want is a generic method to treat Angelina's interaction with her environment, without having to invoke specific microscopic models.

Interaction of the worst (entangling) kind (worst in the sense that these will tend to reduce the off-diagonal elements in Anglina's density matrix, which is where all of her "quantum-ness" resides) happen all the time, between qubits and the environment in which they are embedded: Energy exchange, change of phase, all lead to errors in the qubit evolution. The most general way this kind of error occurs can be understood in the following simple way: We prepare the qubit of interest (Angelina) in a pure state $|A\rangle$. We assume we can write the environment (the universe outside the qubit) as a quantum state $|e\rangle$, initially not entangled with the qubit. If you object that the environment should be described by a density matrix, because we don't know what state it is in, we can use purification of the environment to generate a pure-state description of the environment. See the detailed description that follows if you are interested.

Sidebar: Purification of the environment

The way purification for the environment works is very similar to that for the two-state system discussed earlier. We place the environment in a (for simplicity, at most countably infinite) vector space, spanned by the basis vectors $\{|u_j\rangle, j = 1, \ldots, N\}$. The density matrix is a Hermitian matrix, so we can always find another basis $\{|v_j\rangle, j = 1, \ldots, N\}$ in which the density matrix is diagonal,

$$\rho = \sum_{j=1}^{N} |a_j|^2 |v_j\rangle\langle v_j|. \tag{57}$$

We double the size of the Hilbert space, we span the increased space by adding N additional orthonormal basis vectors $\{|w_j\rangle, j = 1, \ldots, N\}$, then define the extended state of the environment as

$$|e\rangle = \sum_{j=1}^{N} |a_j||v_j\rangle|w_j\rangle = \sum_{j=1}^{N} |a_j||v_j w_j\rangle, \tag{58}$$

which is clearly a pure state. The extended density matrix is then

$$\sigma = |e\rangle\langle e| = \sum_{j,k=1}^{N} |a_j||a_k||v_j w_j\rangle\langle v_k w_k|. \tag{59}$$

To recover the original density matrix, we trace out the additional imaginary states we created to make the state pure,

$$\rho = tr_{w_\ell}(\sigma) = \sum_{j,k,\ell=1}^{N} |a_j||a_k|\langle w_\ell|v_j w_j\rangle\langle v_k w_k|w_\ell\rangle \tag{60}$$

$$= \sum_{j,k,\ell=1}^{N} |a_j||a_k|\delta_{\ell,j}\,\delta_{\ell,k}\,|v_j\rangle\langle v_k| \tag{61}$$

$$= \sum_{j=1}^{N} |a_j|^2 |v_j\rangle\langle v_j|, \tag{62}$$

as desired.

Let's take the outer product of the initial environment state $|e\rangle$ with the qubit state. Let's first assume the qubit starts in the state $|0\rangle$: This gives us the product state $|e\rangle\otimes|0\rangle = |e0\rangle$ (note we're writing the environment state first). Next, we allow these to interact for a short time. The initial environment evolves in this time to $|e_1\rangle$, and the interaction causes an admixture of the orthogonal qubit state $|1\rangle$, with the environment in a different state $|e_2\rangle$. Hence we can write

$$|e0\rangle \Rightarrow a|e_1 0\rangle + b|e_2 1\rangle, \tag{63}$$

The interaction not only changes the qubit, but modifies the environment as well. For a qubit starting instead in the $|1\rangle$ state, we have

$$|e1\rangle \Rightarrow c|e_3 0\rangle + d|e_4 1\rangle. \tag{64}$$

We usually assume the environmental changes are small, so the corresponding error-associated amplitudes are small, i.e. we assume that $|b|, |c| \ll 1$ and that $|a|, |d|$ are just slightly smaller than 1.

We can generalize by using the qubit projection operators P_0 and P_1, written in our $\{|0\rangle, |1\rangle\}$, basis as

$$P_0 = |0\rangle\langle 0| \leftrightarrow \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = \frac{1}{2}(I + Z), \tag{65}$$

written on the right in terms of the Pauli operators, and

$$P_1 = |1\rangle\langle 1| \leftrightarrow \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \frac{1}{2}(I - Z). \tag{66}$$

Using these projection operators we can write the evolution for a general state $|u\rangle = x|0\rangle + y|1\rangle$ as

$$|eu\rangle \Rightarrow (a\,|e_1\rangle I + b\,|e_2\rangle X)\,P_0|u\rangle + (c\,|e_3\rangle X + d\,|e_4\rangle I)\,P_1|u\rangle, \tag{67}$$

where the Pauli operators I and X on the qubit state. The operators P_0, P_1 pull out the appropriate components of $|u\rangle$, and I and X then either leave these as they are, or flip them, $|0\rangle \leftrightarrow |1\rangle$. Note that the notation here is somewhat sloppy, the terms should be written more precisely as e.g. $b\,|e_2\rangle \otimes XP_0|u\rangle$ and so on. The meaning however should be clear.

Equation (67) can be written

$$\begin{aligned} |e\rangle|u\rangle \;\Rightarrow\; & (a\,|e_1\rangle I + b\,|e_2\rangle X)\,\frac{I+Z}{2}|u\rangle + (c\,|e_3\rangle X + d\,|e_4\rangle I)\,\frac{I-Z}{2}|u\rangle \\ = \; & \left(\frac{a|e_1\rangle + d|e_4\rangle}{2} I + \frac{a|e_1\rangle - d|e_4\rangle}{2} Z + \frac{b|e_2\rangle + c|e_3\rangle}{2} X + \right. \\ + \; & \left. \frac{b|e_2\rangle - c|e_3\rangle}{2} XZ \right)|u\rangle. \end{aligned} \tag{68}$$

Now we can write

$$XZ = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = -iY, \tag{69}$$

so we can write this as

$$\begin{aligned} |e\rangle|u\rangle \;\Rightarrow\; & \left(\frac{a|e_1\rangle + d|e_4\rangle}{2} I + \frac{a|e_1\rangle - d|e_4\rangle}{2} Z + \frac{b|e_2\rangle + c|e_3\rangle}{2} X + \right. && (70) \\ + \; & \left. i\frac{c|e_4\rangle - b|e_2\rangle}{2} Y \right)|u\rangle. && (71) \end{aligned}$$

Note that the amplitudes b and c in the terms with the operators X and Y are assumed small. Furthermore, a and d should be nearly equal, and the environmental states $|e_1\rangle$ and $|e_4\rangle$ should be very nearly equal, so the term with operator Z should also be small. We can schematically rewrite this as

$$|eu\rangle \Rightarrow (a_0|e_0\rangle I + \epsilon_x|e_x\rangle X + \epsilon_y|e_y\rangle Y + \epsilon_z|e_z\rangle Z)\,|u\rangle, \tag{72}$$

where the amplitudes $|\epsilon_x|$, $|\epsilon_y|$, and $|\epsilon_z|$, which correspond to X, Y and Z errors in the qubit are all small, and $|a_0| \approx 1$ is very close to unity, which represents no error in the qubit. We see that without many assumptions, and some hand-waving, a general interaction of a qubit with its environment can be written as a no-error outcome, with large amplitude, superposed with small-amplitude environmental interactions that cause X, Z and Y errors. These are known as bit-flip, phase-flip and combined bit-and-phase-flip errors, respectively. Note that further, the amplitude of the errors will scale with the interaction time that the qubit and the environment are interacting, as we saw for the

Angelina-Brad interaction: The longer the qubit and environment interacts, the larger the amplitudes for the X, Y and Z errors, thus the more likely that such errors will occur. If the error amplitudes become too large, then our simple model will break down, because the qubit will begin to suffer multiple errors that generate more complex states.

We now have a handle on what a qubit does when interacting with its environment: We can model short, weak interactions as leaving the qubit state unchanged, superposed with states where the qubit suffered single X, Y and Z errors. These errors leave the qubit entangled with its environment. In principle, if we know what state the qubit is supposed to be in (here $|u\rangle$), we could build a projective measurement operator that projects onto $|u\rangle$, $P_u = |u\rangle\langle u|$, which returns the value 1. This can be combined with its counterpart that projects onto the "not $|u\rangle$" space, $P_{not\ u} = I - P_u$, which returns the value -1. With a probability $|a_0|^2 \approx 1$ we would return the value 1 and the qubit would be returned to its desired state, dis-entangled from the environment; with a probability $1 - |a_0|^2 \ll 1$ we would instead get the value -1, and we would know we have to correct the state of the qubit. However, this is only useful if we know $|u\rangle$, and also only if the qubit is not entangled with other computational qubits. How do we deal more generally with these errors, in a way that doesn't require knowing anything about the state of the qubit, and avoiding having to dis-entangle the qubit from its computational partners?

6 Error correction

Error correction is a fascinating topic in quantum computation. Peter Shor's discovery of a nine-qubit encoding that allows the detection and correction of errors in the encoded qubit state, without knowing the state of the qubit, is a landmark in the evolution of quantum computation. We will introduce this topic by first describing the most simple quantum error-correcting code, a circuit that can detect and correct bit-flip errors without knowing the encoded state.

6.1 Three-bit encoding

We have two parties, Angelina and Brad (these are people now, not qubits) that are going to communicate with one another. Let's assume Angelina and Brad communicate through a classical communication channel, sending zeroes and ones to one another using some kind of binary encoding. However, the communication channel is "noisy" in the sense that a zero can be flipped during transmission to a one, and a one can be flipped to a zero, in some random, unpredictable way. We make the simplifying assumption that this is a *binary symmetric* communication channel, which means that the probability that an error causing $1 \to 0$ is as likely as an error that causes $0 \to 1$, both occurring with probability p; real hardware probably won't respect this simplification, of course. The probability of no error is $1 - p$. We won't consider any other kinds of errors (for example, complete loss of a communication bit, or scrambling the order of a multi-bit communication stream).

Say Angelina and Brad want to lower the probability of a communication error. The simplest way for Angelina to do this is to use a *repetition code*, which just means Angelina will send the information more than once. Brad, upon receiving Angelina's message, can then "vote" the results against one another, with the majority winning. For this to work, Angelina of course needs to send the information an odd number of times. The simplest method is for Angelina to use a *3-bit repetition code*: Angelina encodes each bit of information into three transmission bits, so $0 \to 000$ and $1 \to 111$.

When transmitting this three-bit repetition code, there are a number of things that can happen. One outcome is that none of the bits suffers an error; another possibility

is that one bit suffers an error; or two suffer errors, or all three do. We can write the probabilities for each of these outcomes, assuming the errors are all independent, in other words that there is no correlation between errors. If the probability for one bit to suffer an error is p, then the probability that no bit suffers no error is $(1-p)^3$. The probability that all three bits have errors is p^3; and the probability for one bit or two bits to have errors is $3(1-p)^2p$ and $3(1-p)p^2$, respectively (the factor of 3 because there are three places the error or non-error can occur among the three bits). Note of course these probabilities are just the binomial expansion of $(a+b)^3$ where $a = 1-p$ and $b = p$.

When Bob takes the bits he's received and votes them against one another, a communication error only occurs when there are two- or three-bit errors, so the probability of a communication error goes from p when sending a single bit, to $3(1-p)p^2 + p^3$ when sending three bits. This probability is less than p, if $p < 1/2$; if p is very small, then this error is of order $3p^2 \ll p$. If for example the error probability p is one percent (0.01), then the probabilities of the various errors are shown in the table below. The rate of communication errors falls from one percent to roughly 0.03 percent by using this simple code, a reduction by a factor of 30 at the cost of having to send three times as many bits. Using more bits in the same manner reduces the rate further, in an exponential fashion.

Error syndrome	Probability ($p = 0.01$)
No error	$(1-p)^3 \approx 0.97$
One error	$3(1-p)^2p \approx 0.029$
Two errors	$3(1-p)p^2 \approx 0.0003$
Three errors	$p^3 = 10^{-6}$

Problem 12: Find the communication error rate using a majority voting scheme for a 5-bit encoding, given a one-bit error rate p.

6.2 Bit-flip qubit encoding

Let's now turn to the more interesting problem where Angelina and Brad want to communicate using quantum bits. Let's assume that we can have only bit-flip errors, so we have the error processes $|0\rangle \to |1\rangle$ and $|1\rangle \to |0\rangle$, each occurring with equal probability p. We are thus considering random X gates, occurring with probability p. If the unwanted X gate does not act, then the identity operator I acts instead, with probability $1-p$. Say Angelina wants to send a quantum state $|u\rangle = a|0\rangle + b|1\rangle$. If she sends this state, then there is a probability $1-p$ that Brad will receive the state $|u\rangle$, but there is a probability p that Brad will instead receive the state $X|u\rangle = b|0\rangle + a|1\rangle$, due to the erroneous action of the X operator.

How does Angelina protect against this? She cannot create the state $|u\rangle|u\rangle|u\rangle = |uuu\rangle$, unless she knows the amplitudes a and b. To create the state $|uuu\rangle$, not knowing a and b, Angelina would have to copy the state and thus violate the no-cloning theorem (see sidebar). However, Angelina can instead create and send a kind of state called a *GHZ state*, of the form (for three qubits) $(a|000\rangle + b|111\rangle)/\sqrt{2}$. Here GHZ stands for Greenberger-Horne-Zeilinger, the names of the researchers that explored the utility of such states. Hence Angelina performs this transformation:

$$a|0\rangle + b|1\rangle \Rightarrow a|000\rangle + b|111\rangle = a|\overline{0}\rangle + b|\overline{1}\rangle, \tag{73}$$

where $|\overline{0}\rangle = |000\rangle$, a three-qubit encoding.

Sidebar: No-cloning theorem

A very important theorem in quantum information, the no-cloning theorem, says that it is not possible to copy a quantum state. This means there is no "photocopier", no black-box that can copy quantum states.

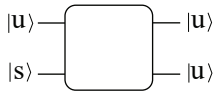

Figure 14: A box that purports to clone the state $|u\rangle$ cannot exist by the no-cloning theorem.

The proof is by negation: Assume there is such a box, so $|u\rangle|s\rangle \rightarrow |u\rangle|u\rangle$ through a unitary transformation (see Fig. 14):

$$U(|u\rangle|s\rangle) = |u\rangle|u\rangle. \tag{74}$$

Now copy another state $|v\rangle$ (the unitary U should work for *any* state):

$$U(|v\rangle|s\rangle) = |v\rangle|v\rangle. \tag{75}$$

Taking the inner product of these two expressions, using the identity $U^\dagger U = I$ and $\langle s|s\rangle = 1$ and remembering that $(AB)^\star = B^\star A^\star$, we have

$$\begin{aligned} (U(|v\rangle|s\rangle))^\dagger U(|u\rangle|s\rangle) &= (|v\rangle|v\rangle)^\dagger |u\rangle|u\rangle \\ \langle v|u\rangle\langle s|s\rangle &= \langle v|u\rangle\langle v|u\rangle \\ \langle v|u\rangle &= \langle v|u\rangle^2 \\ \Rightarrow \langle v|u\rangle &= 1 \text{ or } 0. \end{aligned} \tag{76}$$

Then either $|v\rangle = |u\rangle$, or $|v\rangle$ and $|u\rangle$ are orthogonal. Hence a proposed transformation U that copies a *general* quantum state is impossible; a general U can at most copy one state and its orthogonal partner (but not a superposition of the two).

How would Angelina generate a GHZ state from the state $|u\rangle$? The circuit to do this shown in Fig. 15: Angelina starts with the first qubit in the state $|u\rangle = a|0\rangle + b|1\rangle$, with the accompanying second and third qubits each in the state $|0\rangle$. Alice then does a controlled-NOT using the first qubit as the control and the second qubit as the target, yielding the two-qubit state $a|00\rangle + b|11\rangle$ as the output. She then does a second controlled-NOT with the third qubit, generating the desired three-qubit GHZ state $a|000\rangle + b|111\rangle$.

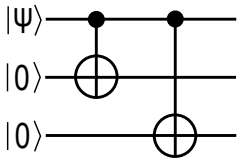

Figure 15: Circuit to extend one-qubit encoding to three-qubit encoding.

Brad receives the state sent by Angelina, but subject to random bit-flip errors on each qubit. He wants to check if there were any errors in transmission. To do this, Brad

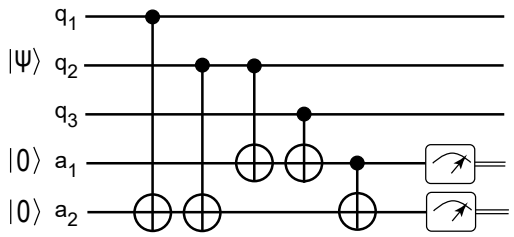

Figure 16: Circuit to generate an error syndrome in two ancilla qubits, using three-qubit encoding. The three communication qubits (q_{1-3}, in the state $|\psi\rangle$) are effectively "voted" against one another using the two ancilla qubits ($a_{1,2}$, both set to $|0\rangle$) at the bottom of the circuit. This kind of circuit does what is called a "parity check."

uses two additional qubits, known as *ancilla* qubits, to "vote" the communication qubits against one another, using the circuit in Fig. 16. This kind of circuit performs what is known as a "parity check".

The way this circuit works is that the first and second qubits control sequential CNOTs on the first ancilla, which effectively makes these two qubits "vote" against one another; if they are not in the same state, the result is that the first ancilla is flipped. Then the second and third qubits do the same to the second ancilla, flipping that ancilla if these two qubits differ. Finally, the two ancillas are compared, thus effectively comparing one and two against two and three, which is the same as one voting against three. The result is that the two ancillas report not only an error, but indicate in binary form where the error occurred; note the circuit only works if at most one qubit suffered an error.

Let's work through this circuit, following its operation step by step. We assume that a single X error has occurred to the second qubit during transmission, so the state Brad receives is $a|010\rangle + b|101\rangle$ instead of the error-free $a|000\rangle + b|111\rangle$. The states following each of the gates in the circuit are then:

$$
\begin{aligned}
|\psi_1\rangle &= (a|010\rangle + b|101\rangle) \otimes |00\rangle && (77)\\
|\psi_2\rangle &= a|010\rangle|00\rangle + b|101\rangle|10\rangle && (78)\\
|\psi_3\rangle &= a|010\rangle|10\rangle + b|101\rangle|10\rangle && (79)\\
|\psi_4\rangle &= a|010\rangle|11\rangle + b|101\rangle|10\rangle && (80)\\
|\psi_5\rangle &= a|010\rangle|11\rangle + b|101\rangle|11\rangle && (81)\\
|\psi_6\rangle &= a|010\rangle|10\rangle + b|101\rangle|10\rangle = (a|010\rangle + b|101\rangle) \otimes |10\rangle. && (82)
\end{aligned}
$$

At the end of the process, the ancilla qubits encode the binary number 10, which indicates an error on the second qubit; the measurement of the two ancillas at the end of this process would identify this error to Brad, and he could then apply a corrective X gate to the second qubit, recovering the state Angelina had sent.

You can work through the other possible errors and see how the circuit works. Table 1 summarizes the outcomes, where q_1 through q_3 are the communicated qubits and A_1 and A_2 are the ancillas; an X in the column for q_j means that that qubit suffered an X error. You can see that, as mentioned above, A_1A_2 is a binary-encoded indicator of where the error occurred. If two or three qubits suffer errors, the circuit does not work, so we didn't tabulate those errors. The ancillas A_1 and A_2 are called an *error syndrome*, indicating what error occurred.

You can work out how to extend this table for what happens with the ancillas for multiple qubit errors.

q_1	q_2	q_3	A_1	A_2
0	0	0	0	0
X	0	0	0	1
0	X	0	1	0
0	0	X	1	1

Table 1: Three-qubit error encoding ancilla results, for single bit-flip errors.

We can now add to the circuit in Fig. 16 so that we use the error syndromes from the two ancillas to automatically correct any single-qubit errors, as shown in Fig. 17, which now includes the error-correcting circuit as well as the error-syndrome circuit.

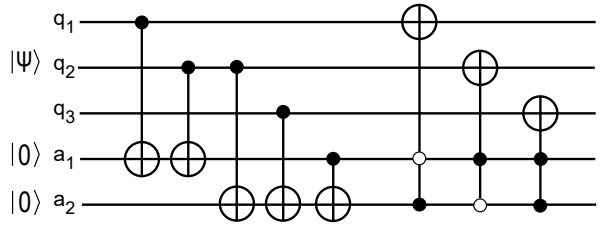

Figure 17: Circuit to detect and correct errors in the three-qubit bit-flip error code. Here we introduce a new set of gates, double-control CNOT gates (these are versions of the Toffoli gate). When the two controls are filled circles, the CNOT is applied when both controls are in the $|1\rangle$ state, while an empty circle means that control qubit must be in the $|0\rangle$ state for the CNOT to act.

This quite interesting circuit unfortunately only corrects bit-flip (X) errors. As we have seen with quantum bits, you can also suffer from Y and Z errors, and as you can check, this circuit does not protect against these. Hence this is not sufficient to correct all single qubit errors.

Problem 13: Work out what happens if a Z error occurs on one of the three qubits in the code. Does the error detection circuit signal an error?

We can build a circuit that detects phase-flip instead of bit-flip errors, by instead encoding the qubits in a GHZ state in the $|\pm\rangle$ basis:

$$a|0\rangle + b|1\rangle \quad \Rightarrow \quad a|000\rangle + b|111\rangle \tag{83}$$

$$\Rightarrow \quad a\,|+++\rangle + b\,|---\rangle. \tag{84}$$

This could easily be implemented by adding a Hadamard to each of the three qubits after the double-CNOT circuit has created the three-qubit GHZ encoding. However, as you can verify for yourself, this circuit no longer protects against bit-flip errors!

6.3 The Shor 9-qubit code

It turns out that (of course) there is a way to protect against both bit-flip and phase-flip errors; this was another of Peter Shor's major discoveries. His code, which was the first example of a fully-protected error code, uses nine qubits in what is called a *concatenated*

code:

$$a|0\rangle + b|1\rangle \quad \Rightarrow \quad a\frac{1}{2\sqrt{2}}\left(|000\rangle + |111\rangle\right)\left(|000\rangle + |111\rangle\right)\left(|000\rangle + |111\rangle\right) \tag{85}$$

$$+ \quad b\frac{1}{2\sqrt{2}}\left(|000\rangle - |111\rangle\right)\left(|000\rangle - |111\rangle\right)\left(|000\rangle - |111\rangle\right). \tag{86}$$

This is called, quite imaginatively, Shor's nine-qubit code. The encoding is done using the circuit shown in Fig. 18. Shor's code has since been improved with a handful of smaller, more efficient codes, and the error-syndrome analysis has been worked out in much greater detail.

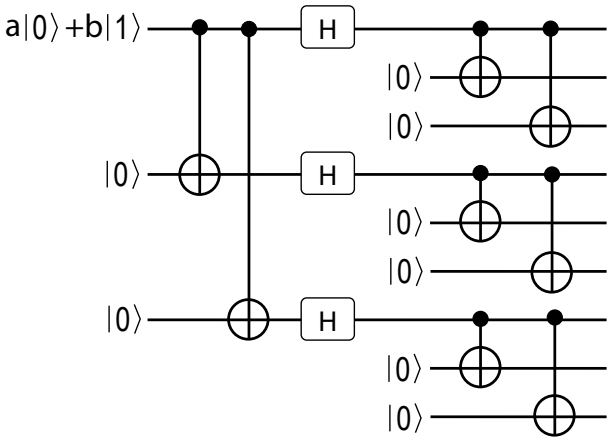

Figure 18: Circuit to generate Shor 9-qubit error-protected state from that of a single qubit.

The way the Shor code works is that each set of three initial qubits in Fig. 18 are first encoded in the bit-flip code $a|000\rangle + b|111\rangle$ that we already looked at. Each qubit in this code is then the basis for a three-qubit block, where a qubit in $|0\rangle$ is encoded as $|{+}{+}{+}\rangle$ and a qubit in $|1\rangle$ becomes $|{-}{-}{-}\rangle$. This allows detection and correction of both bit- and phase-flips. In fact, this code can protect against up to one bit-flip error in each block, and any number of phase-flips in one block. A circuit similar to that shown in Fig. 17 is used to detect X errors in any of the three-qubit blocks; the circuit shown in Fig. 19 is used to detect phase-flip errors in the blocks. The first set of six CNOTs compares the signs of the qubits in blocks 1 and 2, and the second set compares those in blocks 2 and 3. A phase-flip error on any qubit in one of the blocks gives the same outcome, thus the Shor code is known as a degenerate code: Different errors can lead to the same outcome, while in a non-degenerate code each error can be uniquely identified. If the Shor code suffers a Y error, both the X-correcting and Z-correcting codes will work to fix that error, so this is a fully error-protected code.

A number of more sophisticated and compact encoding schemes have been developed since the 9-qubit Shor code. These include the seven-qubit Steane code and a class of five-qubit codes discovered by Ray Laflamme and co-workers. These have been generalized as the CSS codes, named after their inventors, A. Robert Calderbank, Peter Shor and Andrew Steane. There are also a class of stabilizer codes, discovered by Daniel Gottesman and separately by A. Robert Calderbank, Eric Rains, Peter Shor, and Neil Sloane. We now turn to the surface code, which is a kind of stabilizer code.

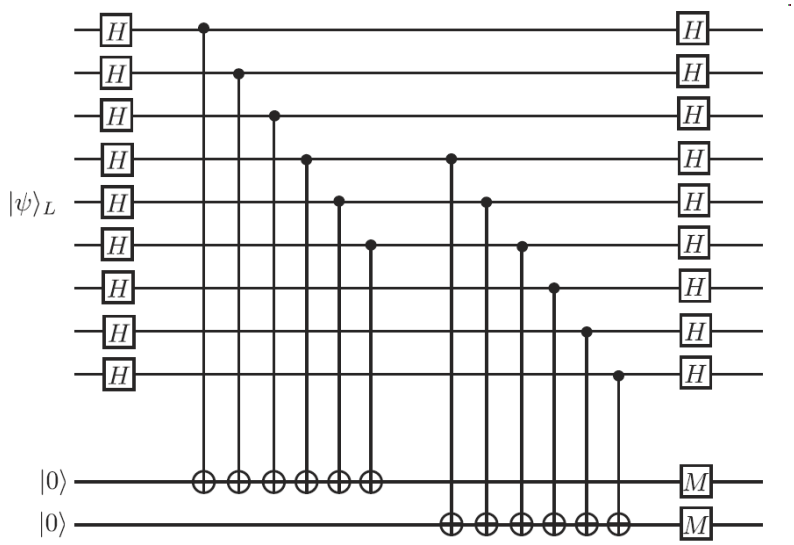

Figure 19: Circuit to detect phase-flip (Z) errors in the Shor 9-qubit codeword.

7 Introduction to the surface code

The surface code is an error-detecting code that embeds some concepts from topology. There are two main parts to the surface code: A geometric layout for the individual physical qubits, and a sequence of operations that are carried out in a continuous cycle with these qubits. The combination of these two parts leads to an error-protected "fabric" in which errors can still occur, but are corrected with exponentially growing probability as the number of physical qubits is increased (as long as the operations that make up the continuous cycle are performed above a certain level of fidelity). Logical qubits can then be defined in this fabric, with size of the logical qubits (the area of the fabric, or equivalently the number of physical qubits involved) determined by the rate at which errors occur in the physical qubits and the rate of errors that can be tolerated at the logical level. The rate at which logical qubit errors can be tolerated depends on the computational problem. If a problem involves say 100 logical operations, and the probability of a logical error is 1%, meaning 99% chance of no error on any operation, then the chance that you can perform all 100 operations without a logical error scales as $0.99^{100} = 0.37$: You have a roughly one-third chance of getting the right answer, which may or may not be tolerable. If instead the calculation involves 1000 operations, then the probability of getting the right answer is $0.99^{1000} \approx 4 \times 10^{-5}$. To recover a one-third chance of no error, the logical error rate would need to be improved by a factor of 10, to 0.1%.

The threshold above which the physical qubit operations need to be performed to obtain the exponential improvement promised by the surface code is around 1%. If the physical qubit operations can be performed with say a $10^{-4} = 0.01\%$ error rate, then it turns out that a logical qubit with an error rate of order 10^{-17} would need about a thousand physical qubits.

The physical layout of the qubits in the surface code is shown in Fig. 20a; the qubits can all be identical, but are used in three different ways, as determined by the cycle of operations. Half of the qubits are "data qubits," represented by open circles in Fig. 20a, storing the quantum state for the computation. The other half are error-detecting qubits, represented by filled circles, which we call "measure qubits." The measurement qubits are used to stabilize and manipulate the quantum state of the data qubits. There are

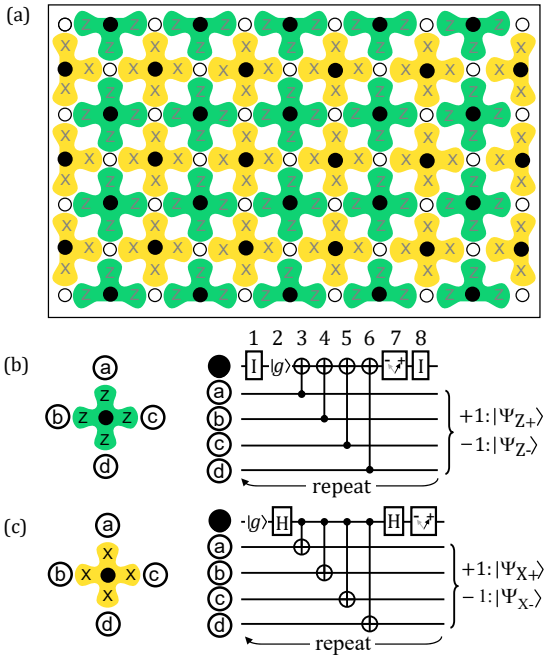

Figure 20: (a) The surface code. Data qubits are open circles (○), measurement qubits are filled circles (●), with measure-Z qubits colored green and measure-X qubits colored orange. Away from the boundaries, each data qubit contacts four measure qubits, and each measure qubit contacts four data qubits; the measure qubits perform four-terminal measurements. On the boundaries, the measure qubits contact only three data qubits and perform three-terminal measurements, and the data qubits contact either two or three measure qubits. The solid line surrounding the array indicates the array boundary. (b) Geometric sequence of operations (left), and quantum circuit (right) for one surface code cycle for a measure-Z qubit, which stabilizes $Z_aZ_bZ_cZ_d$. (c) Geometry and quantum circuit for a measure-X qubit, which stabilizes $X_aX_bX_cX_d$. The two identity I operators for the measure-Z process, which are performed by simply waiting, ensure that the timing on the measure-X qubit matches that of the measure-Z qubit, the former undergoing two Hadamard H operations. The identity operators come at the beginning and end of the sequence, reducing the impact of any errors during these steps. Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

two types of measurement qubits, "measure-Z" qubits, colored green in Fig. 20a, and "measure-X" qubits, colored yellow; these are also called *Z syndrome* and *X syndrome* qubits in the surface code literature; see e.g. Ref. [16]. The cycles for each are shown in Fig. 20b and c. Each data qubit is coupled to two measure-Z and to two measure-X qubits, and each measurement qubit is coupled to four data qubits. In some situations, such as in a logical Hadamard operation, the data qubits are exchanged with the measurement qubits, so every physical qubit must be able to serve in both data and measure roles.

A measure-Z qubit, when performing the measurement cycle shown in Fig. 20b, forces its neighboring data qubits a, b, c and d into an eigenstate of the operator product $Z_aZ_bZ_cZ_d$: Each measure-Z qubit is thus said to measure a Z stabilizer. A measure-X qubit, cycling through the process in Fig. 20c, forces its neighboring data qubits into an eigenstate of $X_aX_bX_cX_d$, and therefore measures an X stabilizer.

8 Quiescent state of the surface code

The measure-Z and measure-X qubits are operated in the cycle shown in Fig. 20b and c. After initializing each measure qubit in its ground state $|g\rangle$, the sequence comprises four CNOTs followed by a projective measurement. For the measure-Z qubit, the CNOTs target the measure qubit with the four nearest-neighbor data qubits as the controls, with the projective measurement yielding an eigenstate of the $Z_aZ_bZ_cZ_d$ product operator. For the measure-X qubit, after the ground state initialization, a Hadamard is applied to the measurement qubit, putting it in $|+\rangle$, the four CNOTs then target the nearest-neighbor data qubits using the measure qubit as the control, and a Hadamard follows this operation; the projective measurement yields an eigenstate of the $X_aX_bX_cX_d$ product operator.

Problem 14: Show that the surface code cycle shown in Fig. 20b for a measure-Z qubit indeed forces the four data qubits into an eigenstate of $Z_aZ_bZ_cZ_d$.

After the projective measurement of all the measure qubits in the array, the state $|\psi\rangle$ of all the data qubits simultaneously satisfies $Z_aZ_bZ_cZ_d|\psi\rangle = Z_{abcd}|\psi\rangle$, with eigenvalues $Z_{abcd} = \pm 1$, and $X_aX_bX_cX_d|\psi\rangle = X_{abcd}|\psi\rangle$ with eigenvalues $X_{abcd} = \pm 1$. This holds for every four-qubit collection *abcd* of data qubits, for the entire surface code array. Following each measurement of the measure-Z and measure-X qubits, the cycle of resets, CNOTs and measurement is repeated. The measure qubits in Fig. 20b and c all operate in lock-step, so that every step in the cycle shown in the figure is completed over the entire 2D array before the next step begins. We note that the particular zig-zag sequence *abcd* for each of the measure qubits is quite particular, and cannot easily be modified while preserving the stabilizer outcomes. It ensures, in part, that no measure-X is interacting at the same time on a particular data qubit as any measure-Z or other measure-X, and similarly for all the measure-Zs. Also, the timing in the surface code sequence shows that measurements of the measure-X and measure-Z qubits happen more or less at the same time, certainly these should all be completed before starting the next surface code cycle, but they do not need to be simultaneous across the array.

The $ZZZZ$ and $XXXX$ measurements, starting with the very first complete measurement cycle when starting up a surface code, take whatever initial state the data qubits are in, and project from that state a complex entangled state $|\psi\rangle$ that is a simultaneous eigenstate of all the measurements across the array; we call this entangled state the *quiescent state*. The particular quiescent state $|\psi\rangle$ that results is selected at random from the enormous number of possible outcomes. Given the measurement outcomes, depending on the detailed structure of the overall array (which we discuss later), there may or may not be more than one possible state for the data qubits; this depends on the number of degrees of freedom left by the set of measurement outcomes.

Once selected, in the absence of errors, the same state $|\psi\rangle$ will be maintained by each subsequent cycle of the sequence, with each measure qubit yielding the same measurement outcome as it measured in the previous cycle, because all the X and Z stabilizer measurements commute with one another. This is trivial for stabilizers that do not have any qubits in common, as single-qubit operators on different qubits always commute. Stabilizers that have qubits in common will always share *two* such qubits, so every X stabilizer shares two data qubits with each neighboring Z stabilizer and vice versa. As we discussed earlier, X and Z measurements that share two qubits in common still commute, even though the individual X and Z operators on one qubit do not commute.

In Table 2 we show the $ZZZZ$ and $XXXX$ eigenstates and their corresponding stabilizer eigenvalues. For any set of four data qubits, measured by say a measure-Z qubit,

the eigenvalue returned by the measure-Z stabilizer restricts the four data qubits to some linear superposition of the eight four-qubit states with the corresponding eigenvalue. The same holds for the four data qubits stabilized by a measure-X qubit. If we consider a set of six adjoining data qubits, two measured only by a measure-Z, two measured by both a measure-Z and a measure-X, and two measured only by a measure-X, we know that the two that are measured by both measure qubits will be in a linear superposition of Bell states, entangled with the other qubits, as the Bell states can be simultaneous eigenstates of both ZZ and XX. The Bell state is not specified, because the nature of the four-fold measurement constrains only the product of the four qubits, not of any pair. How constrained would such a set of qubits be? Each data qubit has two degrees of freedom, the relative amplitude and the phase of its $|0\rangle$ and $|1\rangle$ basis states (i.e. the two angles on the Bloch sphere). With six qubits there are $2 \times 6 = 12$ degrees of freedom. We only have two measurements, which impose four constraints (fixing the real and imaginary parts of the corresponding operator eigenvalue), so we still have eight degrees of freedom.

Eigenvalue	$Z_aZ_bZ_cZ_d$	$X_aX_bX_cX_d$
$+1$	$\|0000\rangle$	$\|++++\rangle$
	$\|0011\rangle$	$\|++--\rangle$
	$\|0110\rangle$	$\|+--+\rangle$
	$\|1100\rangle$	$\|--++\rangle$
	$\|1001\rangle$	$\|-++-\rangle$
	$\|0101\rangle$	$\|+-+-\rangle$
	$\|1010\rangle$	$\|-+-+\rangle$
	$\|1111\rangle$	$\|----\rangle$
-1	$\|0001\rangle$	$\|+++-\rangle$
	$\|0010\rangle$	$\|++-+\rangle$
	$\|0100\rangle$	$\|+-++\rangle$
	$\|1000\rangle$	$\|-+++\rangle$
	$\|0111\rangle$	$\|+---\rangle$
	$\|1011\rangle$	$\|-+--\rangle$
	$\|1101\rangle$	$\|--+-\rangle$
	$\|1110\rangle$	$\|---+\rangle$

Table 2: Eigenstates for the four-qubit stabilizers $Z_aZ_bZ_cZ_d$ and $X_aX_bX_cX_d$.

Consider now a larger piece of the surface code, such as that in Fig. 21. This has been drawn with two types of boundaries, terminating with measure-X qubits on the right and left, which we call X boundaries, and measure-Z qubits on the top and bottom, which we call Z boundaries. The X boundaries are also called *smooth boundaries* and the Z boundaries are called *rough boundaries* [16].

If we count the number of data and measure qubits in the array, we find there are 41 data qubits and 40 measure qubits, so there are 2×41 degrees of freedom (with 2^{41} distinct states) in the data qubits, with 2×40 constraints from the stabilizer measurements, which are all linearly independent. To see this, you need to realize that each stabilizer measurement picks one of two possible outcomes (± 1) so reduces the number of degrees of freedom by a factor of two. Hence there are net two unconstrained degrees of freedom in this piece of surface code array: This is equivalent to a qubit, and in fact, this patch of the surface code forms a logical qubit, as we will see later. If we had instead drawn this array with only X or only Z boundaries, instead of half X and half Z boundaries, then it turns out that one of the Z or X stabilizers can be written as a product of all the

other Z or X stabilizers. As the stabilizers all commute, this means that this stabilizer does not impose an additional constraint, so with N measure qubits you only have $2N-2$ constraints. However, in these arrays (as you can easily convince yourself) there are only $N-1$ data qubits, so in these arrays there are no additional degrees of freedom: The stabilizer measurements completely determine the quiescent state, there is no freedom given a set of measurement outcomes, so such patch of the surface code array could not serve as a logical qubit.

9 Manipulating a logical qubit

As we have just discussed, the segment of surface code array shown in Fig. 21 has two unconstrained degrees of freedom. Thus we can recast the state $|\psi\rangle$ of the array into the outer product of a logical qubit state $|u\rangle_L$ with the fully stabilized state $|v\rangle$ of the rest of the array. In other words, we can write

$$|\psi\rangle = |u\rangle_L \otimes |v\rangle, \tag{87}$$

where $|v\rangle$ lives in a 2^{40} dimensional vector space and is constrained to a single state by the stabilizer measurements. We now want to find logical operators X_L and Z_L that can manipulate $|u\rangle_L$ without affecting $|v\rangle$.

Consider the operators $X_L = X_1X_2X_3X_4X_5$ and $Z_L = Z_6Z_7Z_3Z_8Z_9$, shown in Fig. 21. Note that X_L operates on a chain of qubits that connect the two X boundaries of the array, and Z_L operates on a chain that connects the two Z boundaries, which is important. As drawn, these operators commute with every stabilizer in the figure: X_L trivially commutes with all the X stabilizers, and because it bit-flips (applies X_j to) either none, or two, of the four qubits stabilized by each Z stabilizer, it also commutes with all the Z stabilizers. Applying X_L to the array thus does not change any of the X or Z stabilizer measurements. The same holds for Z_L, which phase-flips (applies Z_j to) none or two of the four qubits in each X stabilizer, so commutes with all the X stabilizers, and trivially commutes with all the Z stabilizers. However, these operators clearly change the state $|\psi\rangle$ of the array, because X_L bit-flips five qubits, and Z_L phase-flips five qubits. Furthermore, we can easily show that $X_LZ_L = -Z_LX_L$, so these obey the commutator relations for X and Z operators on a qubit. We thus have defined two operators that manipulate the independent degrees of freedom of our logical qubit $|u\rangle_L$, without affecting the stabilized state $|v\rangle$.

Problem 15: Prove that $X_LZ_L = -Z_LX_L$ by commuting the individual qubit operators that make up these logical operators.

Problem 16: Try defining X'_L and Z'_L as chains of X_j and Z_j operators that link the Z and X boundaries respectively (i.e. perpendicular to the chains we defined for X_L and Z_L). Why doesn't this work?

It may seem that we could choose other chains of single qubit operator products to define different X_L or Z_L operators. Consider $X'_L = X_1X_{10}X_{11}X_{12}X_3X_4X_5$, shown in Fig. 21. This satisfies the same conditions as X_L, as the X_j data qubit operators are paired so as to bracket each measure-Z qubit, so X'_L commutes with all the stabilizers, and will generate a quiescent state $|\psi_{X'}\rangle = X'_L|\psi\rangle$ with the same measurement outcomes as $|\psi\rangle$ and $|\psi_X\rangle$. However the state $|\psi_{X'}\rangle$ is actually trivially related to $|\psi_X\rangle$: First, note

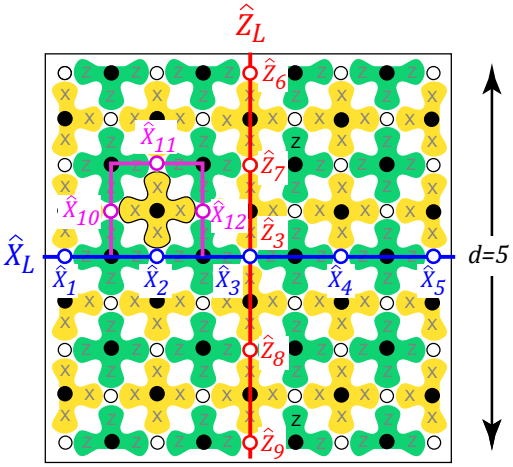

Figure 21: A square 2D array of data qubits, with X boundaries on the left and right, and Z boundaries on the top and bottom. The array has 41 data qubits, but only 40 $XXXX$ and $ZZZZ$ stabilizers. A product chain $X_L = X_1X_2X_3X_4X_5$ of X_j operators connects the two X boundaries and commutes with all the array stabilizers . The product chain $Z_L = Z_6Z_7Z_3Z_8Z_9$ connects the two Z boundaries and also commutes with the array stabilizers. A modification of the X_L chain to the chain $X'_L = X_1X_{10}X_{11}X_{12}X_3X_4X_5$ generates a quiescent state $|\psi_{X'}\rangle = X_{2,10,11,12}|\psi_X\rangle$, equal to ±1 times $|\psi_X\rangle$, as determined by the result of the measurement $X_{2,10,11,12} = \pm1$ of the encircled measure-X qubit (outlined in black). Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

that we can manipulate the set of operators appearing in X'_L, as

$$\begin{aligned} X'_L &= X_1X_{10}X_{11}X_{12}X_3X_4X_5 \\ &= (X_2X_{10}X_{11}X_{12})\,(X_1X_2X_3X_4X_5) \\ &= (X_2X_{10}X_{11}X_{12})\,X_L, \end{aligned} \tag{88}$$

where we have used $X_2^2 = I$. The operator in parentheses in the third line of Eq. (88) is the stabilizer outlined in black in Fig. 21: We have shown that X'_L is just X_L multiplied by an operator product that is stabilized to one of its ±1 eigenvalues by the surface code stabilizers. If we operate on a quiescent state $|\psi\rangle$ with X'_L, we can simply replace this operator product by its eigenvalue:

$$X'_L|\psi\rangle = X_{2,10,11,12}X_L|\psi\rangle = \pm X_L|\psi\rangle. \tag{89}$$

Hence X'_L is trivially related to X_L. This holds for *any* operator X'_L that can be written as a stabilized operator product times X_L. *Any* X'_L chain that crosses the array in Fig. 21 can be written as X_L multiplied by a product of $XXXX$ stabilizers: There is only one non-trivially distinct X_L operator for this array.

This result may seem odd: If we operate with a loop of $XXXX$ operators corresponding to a stabilizer, we are bit-flipping four data qubits, so the state $XXXX|\psi\rangle$ should be different from $|\psi\rangle$. However, the fact that $XXXX$ is equal to a stabilizer means that $|\psi\rangle$ already includes superposition states of the un-bit-flipped and the bit-flipped states of the data qubits involved in $XXXX$, so the loop operators is essentially just swapping states already in $|\psi\rangle$, with no net effect.

The same argument applies to alternative Z_L' operators: Any such operator is equal to the original Z_L multiplied by a product of $ZZZZ$ stabilizers, and thus any $Z_L'|\psi\rangle$ differs from $Z_L|\psi\rangle$ by ± 1. Hence there is also only one linearly independent Z_L operator for this array.

Our array only gives us two degrees of freedom unconstrained by the stabilizers, and only defines a single logical qubit. Even if we make the array larger, while maintaining the half-X and half-Z boundaries, the array will only have two degrees of freedom, so won't allow us to define more logical qubits. Later, we will show how to make larger numbers of logical qubits within an array, by adding more degrees of freedom within the array by turning off stabilizer measurements. Each logical qubit increases the size of the logical Hilbert space by a factor of two, and concomitantly reduces the size of the stabilizer Hilbert space of by a factor of two.

10 Error detection

We now consider errors in the surface code. Possible errors include erroneous single qubit gates, measurement errors, initialization errors, and CNOT errors. These can comprise concatenated errors, which involve *by chance* multiple neighboring data qubits; such concatenated errors are called *error chains*. We do not consider here long-range correlated errors, i.e. errors which link distant physical qubits. These are tolerated to some extent in the surface code, meaning in some cases they can be corrected, but depending on the details, these can also lead to serious logical errors. Single qubit errors, and local chains of errors, as long as they can be identified correctly, can be corrected for by tracking them and correcting any subsequent measurement outcomes using classical control software. Edmonds' minimum-weight perfect-matching algorithm [55, 56] provides an automated method for doing this, and works perfectly for sufficiently sparse errors, but begins to fail as the error density increases, and as the length of the error chains increases. Numerical simulations provide estimates of the tolerance of the surface code to different types of errors, such as shown in Fig. 22.

The simulations used to generate Fig. 22 include the following types of errors, occurring during the surface code cycle shown in Fig. 20:

1. Attempting to perform a data qubit identity I, but instead performing a single qubit operation X, Y, or Z; the probability for any one of these is $p/3$.

2. Attempting to initialize a qubit to $|0\rangle$, but instead generating $|1\rangle$ with probability p.

3. Attempting to perform a measure qubit Hadamard H, but performing in addition one of X, Y, or Z, each with probability $p/3$.

4. Performing a measure qubit Z measurement, but reporting the wrong value and projecting to the wrong state with probability p.

5. Attempting to perform a measure qubit-data qubit CNOT, but instead performing any one of the fifteen two-qubit operations $I \otimes X, \ldots, Z \otimes Z$, each with probability $p/15$ (only fifteen because $I \times I$ can be ignored).

The errors occur randomly during the simulation. The probability p is per step in the surface code cycle (there are eight steps in the cycle), so the overall rate per cycle of the surface code is approximately $8p$. Edmonds' matching algorithm maps changes detected in the stabilizer outcomes to physical qubit errors; the rate at which the algorithm makes mistakes, meaning it misidentifies the source of a particular error report, is displayed as

the logical error rate P_L, defined as the number of logical X_L errors appearing anywhere in the array, per surface code cycle. There are of course also Z_L errors, which occur at about the same rate as X_L errors, and for which the scaling with p is very similar; here we only look at X_L errors.

The relation between P_L and p depends very strongly on the array size d, termed the *distance* of the array: d is the minimum number of physical qubit bit-flips or phase-flips needed to define an X_L or Z_L operator. In Fig. 21, for example, we need a minimum of five physical operators to define a logical operator, so that array has a distance $d = 5$.

For small physical error rates p, the logical error rate P_L is small, and gets smaller as the array distance d increases. For large p, P_L is larger, and gets larger as d increases. The cross-over between these two regimes occurs when p crosses the threshold error rate $p_{\rm th}$: For $p < p_{\rm th}$, the logical error rate falls approximately exponentially with d, while for $p > p_{\rm th}$, P_L increases with d. In Fig. 22, the threshold rate is $p_{\rm th} = 0.57\%$.

For error rates $p < p_{\rm th}$, the simulations show that the logical error rate scales with p according to the power law $P_L \sim p^{d_e}$, where we define the error dimension d_e for odd d as

$$d_e = (d+1)/2. \tag{90}$$

For even d, we round down, so $d_e = d/2$. Using this, the error rate P_L is approximately

$$P_L \cong 0.03\ (p/p_{\rm th})^{d_e}\,. \tag{91}$$

10.1 Statistical model for the logical error rate

The error rate scaling can be understood using a simple model. In Fig. 23a we show two measure-Z qubits reporting errors, marked by the letter E. Two possible sets of data qubit errors that could generate this error are shown in Fig. 23b and c: Either two data qubits had X errors (b), or the other three data qubits had X errors (c). These events look identical to the measure qubits. In general, for a size $d \times d$ array, n actual errors can look like $d-n$ errors and be misidentified as such; the error-identifying algorithm will make this mistake if $d-n$ is smaller than n, as $d-n$ errors are more likely. For an array with distance d, with d odd, the most identification mistakes of this type occur when $d_e \equiv (d+1)/2$ errors occur and are mistakenly identified as $(d-1)/2$ errors. The rate this occurs scales as p^{d_e}. This explains the general scaling seen in Fig. 22 and Eq. (91). This underlines the primary result that in the surface code, misidentification of qubit errors causes logical errors, and that large d arrays are thus less error-prone than small arrays (when operating below the error rate threshold).

The logical error rate can be estimated using statistical arguments. Chains of errors will give the same measurement outcomes if they are complementary, as in Fig. 23: Two chains of errors are complementary if their product is an array-crossing chain that commutes with all the stabilizers (in other words, the complementary chains' product is a logical operator, or equivalent to a logical operator). Since shorter error chains are more likely, we only consider minimum-length chains that cross the array. These have d operators, and can cross the array in any one of the d data qubit rows. The most likely mistakes occur for error chains with length $d_e - 1 = (d-1)/2$ that are complementary to error chains of length $d_e = (d+1)/2$. For a chain of errors in a given row, the number of possible d_e-fold errors is $d(d-1)\ldots d_e/d_e!$. Given a per-step error rate p, the per-cycle individual error rate is $p_e \cong 8p$, as there are eight steps per cycle. The total X_L error rate P_L from these statistical arguments is then

$$P_L = d \times \frac{d!}{(d_e-1)!d_e!} \times p_e^{d_e}, \tag{92}$$

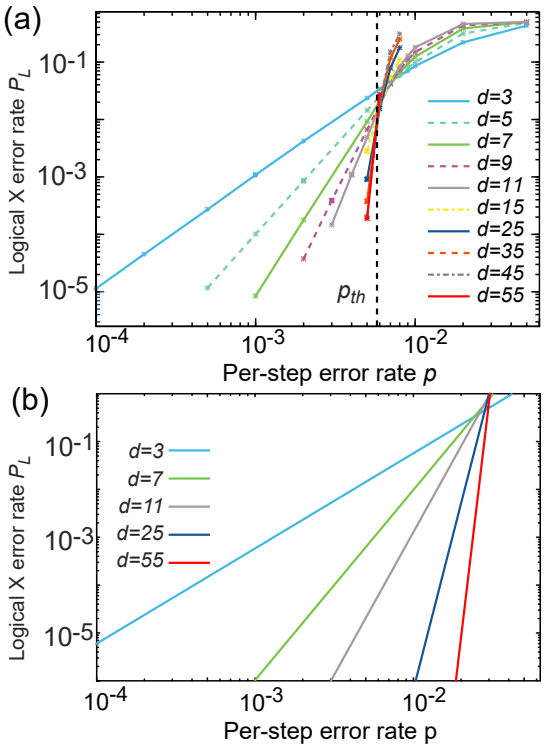

Figure 22: (a) Surface code logical error rate P_L as a function of per-step error rate p, for different distance d square arrays. Per-step error rates p that are below the threshold error rate $p_{\rm th} = 0.57\%$ (dashed line) yield surface code logical error rates P_L that scale roughly as p^d. The threshold corresponds to roughly a $0.0057 \times 8 \approx 4\%$ error rate for the entire surface code measurement cycle. (b) Estimated error rates, using statistical arguments given in main text, for array distances $d = 3$, 7, 11, 25 and 55. Simple estimate is qualitatively similar to detailed simulations. Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

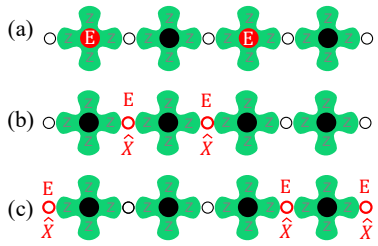

Figure 23: (a) Two measure-Z qubits report errors (E) in a row of a 2D array. This could be generated by (b) two X errors appearing in one surface code cycle, or (c) three X errors appearing in the other three data qubits. Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

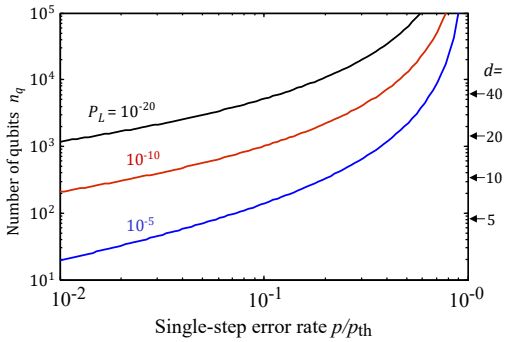

Figure 24: Estimated number of physical qubits n_q per logical qubit as a function of the per-step error rate p, normalized to the threshold error rate p_{th}. This is plotted for different logical error rates P_L. Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

where the factor of d accounts for the d independent rows in the array. A plot of this prediction is shown in Fig. 22b, and scales in the same way as the more complete simulations in Fig. 22a.

We can estimate the number of qubits needed to obtain a desired logical error rate. Using Eq. (91) for the error probability, we plot in Fig. 24 the total number n_q of data and measurement qubits, $n_q = (2d-1)^2$, versus the per-step error rate normalized to the threshold $p/p_{\text{th}} < 1$, for three values of the logical error rate P_L. We find that n_q increases rapidly as p approaches the threshold p_{th}. For a uniform gate fidelity of 99.9% ($p = 10^{-3}$), we have $p/p_{th} = 0.18$. In this case, a logical qubit will need to contain $10^3 - 10^4$ physical qubits in order to achieve logical error rates below $10^{-14} - 10^{-15}$, just barely sufficient to perform Shor's algorithm for a 500-digit number with a reasonable chance of success.

A similar discussion applies to errors occurring in time instead of the spatially-correlated errors we have discussed so far. A particular pattern of errors in time will suffer misidentifications with the same scaling as in Eq. (92), so we need roughly the same distance d in time (measured in complete surface code cycles) as we do in space to obtain the same logical error rate. Hence a $d = 5$ array needs to execute the surface code cycle five times to achieve the same error distance in time. See Ref. [57].

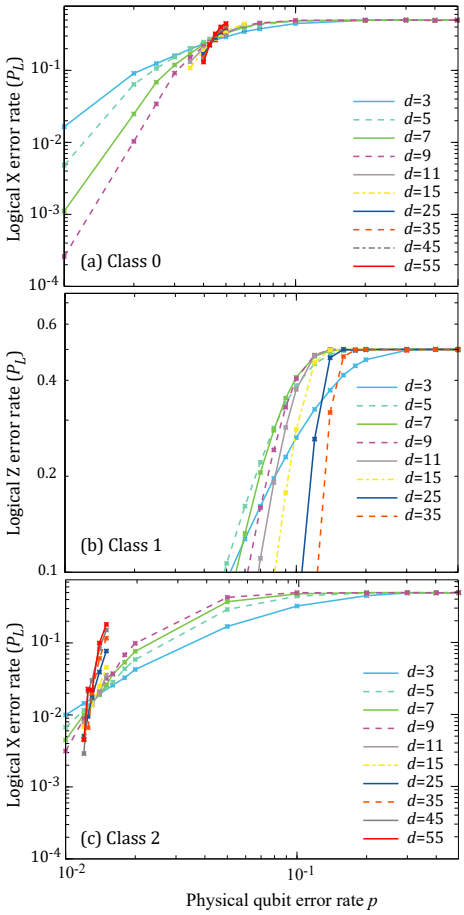

Figure 25: Logical X_L error rate P_L per surface code cycle as a function of the per-step physical error rate p, considering only (a) class-0 (data qubit) errors, (b) class-1 (measure qubit) errors and (c) class-2 (CNOT) errors. Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

10.2 Logical error rate for different error classes

Logical errors occur more frequently for some types of errors than for others. We can classify errors by their approximate impact on the logical error rate. Errors on the data qubits, primarily identity ("idle") operations disturbed by erroneous X, Y or Z operations, are called "class-0" errors. There are four idling steps per surface code cycle. Errors that occur on the measure qubits, namely initialization, measurement, and Hadamard operations, are called "class-1" errors. Each measure-X qubit has four possible class-1 errors per cycle, and each measure-Z two possibilities. There is thus an average of three opportunities per surface code cycle for a class-1 error. Finally, errors in the measure qubit-data qubit CNOT operations are called "class-2" errors. There are four class-2 error opportunities per surface code cycle.

Simulations showing the logical error rate P_L calculated for each error class separately (assuming the other error classes do not occur) are shown in Fig. 25. The logical error rate is most sensitive to class-2 errors, which have a per-step threshold of 1.25% (per-cycle threshold of $\sim 5\%$): CNOTs need to be performed accurately.

The dependence of the overall logical error rate on the three classes of errors occurring concurrently is shown in Fig. 26, where we display the contours of error rates (p_0, p_1, p_2)

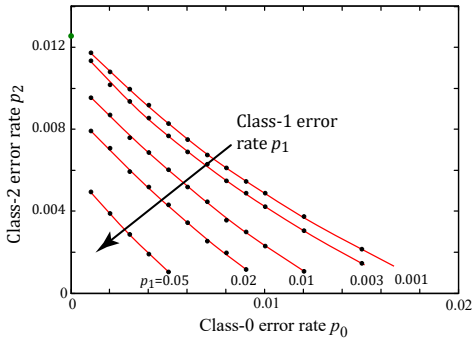

Figure 26: Contours of the three classes' error rates p_0, p_1 and p_2 that combine to give the error threshold, corresponding to the total logical error rate $P_L = 0.02$. Data are from a distance $d = 9$ numerical simulation, and lines are guides to the eye. The green dot (upper vertical axis) is the class-2 error rate giving $P_L = 0.02$ with no class-0 or class-1 errors. Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

that give a total logical error rate $P_L = 0.02$, approximately the value of P_L at threshold. This figure shows that class 0 and 2 errors have roughly equal impact on P_L, while P_L is roughly five times less sensitive to class 1 errors.

Returning to the general error rate discussion, we see that the surface code is able to handle a per-step physical qubit error rate up to the threshold $p_{\text{th}} = 0.57\%$, while still preserving the integrity of logical states in the array.

11 Creating logical qubits

We know how to create a logical qubit from a $d \times d$ "swatch" of the surface code, and we have shown how to build the distance d logical X_L and Z_L operators as array-crossing chains (products) of individual qubit X and Z operators, with X_L and Z_L commuting with all the stabilizers. However, this construct only gives us a single logical qubit. To add more logical qubits, we need to recognize that the qubit degrees of freedom come from the boundaries in the surface code; for our "swatch," the outer boundary provides the needed freedom. However, we can create more boundaries *interior* to the swatch, in other words we can create holes within the fabric of the surface code. The holes, known as *defects* in the published literature, are created by simply turning off one or more of the internal measure-X and measure-Z qubits - here, "turning off" means that the measure qubits inside a hole no longer perform their stabilizer measurement cycles; the qubits are not physically removed or otherwise disturbed. As long as we preserve the required error-protecting distance d, we can create any number of logical qubits in this way, each with their respective logical operators. We will see later that these "internal" qubits further offer a way to create a logical controlled-NOT gate that is itself protected by the surface code stabilizer cycle.

In Fig. 27 we show one way to do this, where we turn off a single measure-Z qubit, thereby creating a hole. As the Z-cut stabilizer no longer performs its surface code cycle, releasing the constraint on the product $Z_a Z_b Z_c Z_d$ of its neighboring data qubits, we have created two additional degrees of freedom in the surface code array. These degrees of freedom can be manipulated in a way similar to the way we manipulate our previous array qubit, by defining logical operators X_L and Z_L that anti-commute. The hole we call a

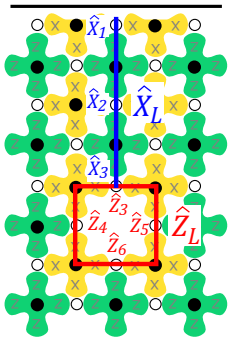

Figure 27: A single Z-cut qubit. The array boundary is indicated by a solid black line (top), while the other three sides of the array continue outwards indefinitely. Two logical operators, $X_L = X_1X_2X_3$ from the array's outer X boundary to the internal X boundary of the Z-cut hole, and $Z_L = Z_3Z_4Z_5Z_6$ surrounding the Z-cut hole, are shown. Note that the operator chains for X_L and Z_L have one physical data qubit in common (data qubit 3). Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

"Z-cut hole", and the qubit that corresponds to it is called a "Z-cut qubit".

In the figure, the Z-cut hole is near an X boundary of the array. The hole is surrounded by measure-X qubits, in other words making the Z-cut hole has created an internal X boundary. We define a logical X operator as $X_L = X_1X_2X_3$, which connects the array's outer X boundary with the internal X boundary of the hole. This logical operator is a chain of operators that are paired across each Z stabilizer, so it commutes with all the Z stabilizers in the array, and trivially commutes as well with the X stabilizers. We also define a logical Z operator as $Z_L = Z_3Z_4Z_5Z_6$, which comprises a loop of Z operators surrounding the Z-cut hole. These Z operators are paired across each X stabilizer, so commutes with each of these, and trivially with all the Z stabilizers as well. This operator loop is not constrained to an eigenstate of a Z stabilizer as it would be without the Z-cut hole, so this operator can change the quiescent state $|\psi\rangle$ of the array in a non-trivial way.

The X_L chain and the Z_L loop share one data qubit, number 3 in Fig. 27. These two operators therefore anti-commute, as you can easily prove to yourself. You can also easily show that $X_L^2 = Z_L^2 = I$, and we can define $Y_L = Z_LX_L$. We have thus created a set of anti-commuting single-qubit Bloch operators associated with this Z-cut hole, so the Z-cut hole is a logical qubit. We note that if the outer perimeter of the swatch of surface code were a single Z boundary, there would be no way to build an appropriate X_L chain for the Z-cut hole. Equivalently, the product of all the Z stabilizers in the array would fix the value of Z_L, so creating a Z-cut hole in a Z-boundary swatch would not create any additional degrees of freedom: At least one X boundary is needed. Note the distance d of this qubit is $d = 3$, limited by the length of the X_L chain from the hole to the swatch boundary. The distance could be increased to $d = 4$ by moving the hole one stabilizer cell further away from the array boundary. However d would then be limited to 4 by the size of the Z-cut hole, which limits the Z_L operator to a product of four operators. Turning off more Z stabilizers in the hole would make it possible to have a larger distance qubit. A single Z-cut qubit is called a *smooth defect* or a *dual defect* in the literature.

An X-cut qubit can be formed in an analogous way, by turning off a single measure-X qubit in an array that has at least one Z boundary; this logical qubit's Z_L operator is then a chain of Z operators from the array Z boundary to the internal Z boundary created by turning off the measure-X qubit, and the X_L operator is the loop of X bit-flips surrounding

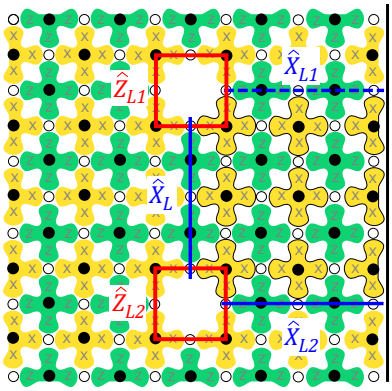

Figure 28: A double Z-cut qubit. Each Z-cut hole has the logical operators X_{L1} and Z_{L1}, X_{L2} and Z_{L2}. We can replace X_{L1} (dashed blue line) by $X_{L1}X_{L2}$, which with X_{L2} is equivalent to the two separate X operators. However, $X_{L1}X_{L2}$ can be multiplied by all the outlined X stabilizers, which at most give a sign change to the operator, to yield the equivalent operator X_L (solid blue line) that links the two qubit holes. Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

the X-cut hole. An X-cut qubit is called a *rough defect* or a *primal defect* in the published literature.

This concept can be made even more general by creating the qubits in pairs, i.e. creating two Z-cut or two X-cut qubits. The logical operators for these paired qubits then are created by a chain of operators linking the qubits (an X chain for Z-cut qubits, a Z chain for X-cut qubits), combined with a loop of operators surrounding the hole (a Z loop for a Z-cut qubit, and an X loop for an X-cut qubit). An example of a "double Z-cut qubit," achieved by turning off two measure-Z qubits in a swatch, is shown in Fig. 28. Making these two Z-cut holes adds four additional degrees of freedom to the array. We define X_{L1} and Z_{L1} for the upper (first) Z-cut hole, and X_{L2} and Z_{L2} for the lower (second) Z-cut hole, as shown in the figure. These four linearly independent operators manipulate all four degrees of freedom, with each logical operator pair (X_{Lj}, Z_{Lj}) commuting with the other pair but anti-commuting with each other.

We can manipulate the two qubit holes together by replacing X_{L1} by the product $X_{L1}X_{L2}$; this anti-commutes with both Z_{L1} and Z_{L2}, and together with X_{L2} provides the same functionality as X_{L1} and X_{L2}. Now we define a new X_L operator linking the two qubit holes, shown in Fig. 28; this is the product of three data qubit X operators, and links the internal X boundary of the upper qubit to the internal X boundary of the lower qubit. Note that $X_L X_{L1} X_{L2}$ is equal to the product of all the stabilizers enclosed by these operators (these stabilizers have black outlines in Fig. 28). Hence, if we multiply $X_{L1}X_{L2}$ by all these stabilizers, the result is X_L, so X_L is equivalent to $X_{L1}X_{L2}$ to within ± 1 (the sign is equal to the product of all the enclosed stabilizers). Hence we can replace $X_{L1}X_{L2}$ by X_L.

We now have the operator set $\{X_L, X_{L2}, Z_{L1}, Z_{L2}\}$, where the logical X operators commute with one another, as do the logical Z operators, but X_L anti-commutes with both Z_{L1} and Z_{L2}; X_{L2} anti-commutes only with Z_{L2}.

However, we are only interested in using two of the degrees of freedom in the double qubit, the two that can be manipulated with "local" operator chains (ones that do not extend to the swatch edge). We therefore use X_L along with either Z_{L1} or Z_{L2}. If we choose to use $Z_{L2} \equiv Z_L$, and define $Y_L = Z_L X_L$, we have a complete set $\{X_L, Z_L, Y_L\}$

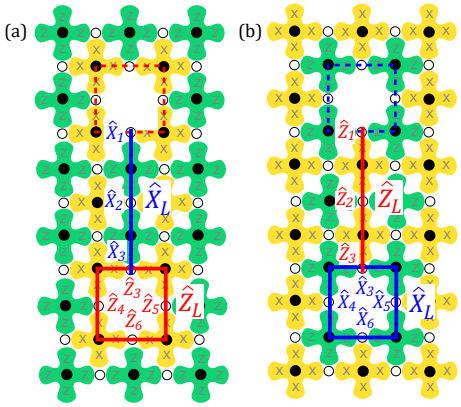

Figure 29: (a) A double Z-cut and (b) a double X-cut qubit, formed by turning off two measure-Z and two measure-X qubits, respectively. For the double Z-cut qubit the logical operators are $X_L = X_1X_2X_3$, a chain that links one Z-cut hole's internal X boundary to the other, and a $Z_L = Z_3Z_4Z_5Z_6$ loop that encloses the lower Z-cut hole. For the X-cut qubit, $Z_L = Z_1Z_2Z_3$, a chain that links the two internal X-cut holes' Z boundaries, and $X_L = X_3X_4X_5X_6$ loop that encloses the lower X-cut hole. X_L and X_L share one data qubit (qubit 3), so the operators anti-commute. Note that the loop operators can be defined to surround either of the two holes in the qubit. Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

of logical qubit operators. By restricting ourselves to this set of logical operators, we are manipulating half of the degrees of freedom provided by the two qubit holes. Hence effectively we are manipulating a single logical qubit. We write the state of this logical qubit in the usual way, as e.g. $\alpha|g_L\rangle + \beta|e_L\rangle$. The advantage this concept provides is that now our double Z-cut qubit can be manipulated using only local operator chains: We are not longer required to build chains that link the qubit to the edge of the surface code patch. This will be very significant when we look at braiding operations between pairs of qubits.

We can create a double X-cut qubit in a similar fashion, by turning off two measure-X qubits. Both the double Z-cut and double X-cut qubit types are shown in Fig. 29. The logical operators for the double X-cut qubit are defined in a similar way: Z_L is a chain linking the two X-cut holes, and either $X_L = X_{L1}$ or $X_L = X_{L2}$, the operator loops that surround one or the other of the X-cut holes. It is easy to show that these anti-commute.

As we will see later, the two different types of logical qubits (X-cut and Z-cut) are both needed to perform the topological braid transformation that provide the logical CNOT operation in the surface code: Only braids between mixed qubit types give the needed functionality. However, as we discuss below, a topological CNOT can be performed between two Z-cut qubits, by using an X-cut qubit as an intermediary; similarly, a CNOT between two X-cut qubits can be performed using a Z-cut as an intermediary. An arbitrary quantum computation can therefore be completed using mostly one flavor of logical qubit, with the other qubit type only making appearances in a supporting role.

The error-protection distance for the logical qubits we've been discussing is only $d = 3$, meaning these are relatively fault-intolerant. Moving the two holes further apart by one surface code unit cell increases the distance to $d = 4$, limited by the length of the Z_L or X_L loops; increasing the size of the holes in parallel with the hole distance will increase the

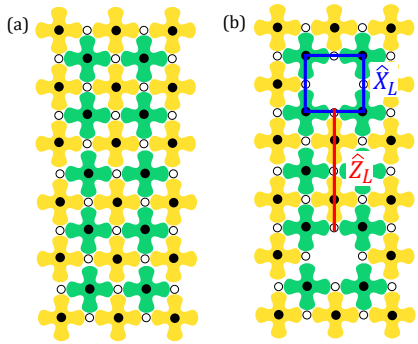

Figure 30: "Easy" initialization of an X-cut logical qubit, where the stable measure-X outcome just prior to creating the qubit hole is equal to the initial logical eigenvalue of the qubit. Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

distance d further, as long as measurements are repeated d times in time in tandem with the increased physical distance. The repetition in time provides protection from errors that occur in the measure-Z and measure-X qubit measurement cycles, where e.g. an isolated error in the measure-Z surface code cycle would be detected by repeated measurement cycles, providing a distance d in time equivalent to the distance d in space.

12 Logical qubit initialization and measurement

How does one initialize and measure a logical qubit? We describe two methods to initialize the logical state, and then two methods to measure the state.

12.1 Initialization.

There are two distinct ways to initialize a logical qubit state: The "easy" way and the "difficult" way. The easy method is to initialize the qubit in an eigenstate of the qubit cut, so for an X-cut qubit, easy initialization is in the $|+_L\rangle$ or $|-_L\rangle$ eigenstates of X_L, while for a Z-cut qubit, easy initialization is in the $|0_L\rangle$ or $|1_L\rangle$ eigenstates. This is illustrated for an X-cut qubit in Fig. 30, starting with a 2D array with no cuts, and stepping immediately to one with a double-X cut qubit, achieved by turning off two measure-X qubits to create two X-cut holes. If we define X_L as the loop surrounding the upper qubit hole, the initial state of the qubit is the measurement outcome of the upper measure-X qubit just before it was turned off (sampled and voted d times to provide error protection). If the initial state is not the desired one, we can apply a Z_L phase-flip to get the desired state.

The alternative is to perform a "difficult" initialization of the X-cut qubit in the Z_L basis, i.e. to initialize the qubit in either $|0_L\rangle$ or $|1_L\rangle$. This can be done using a logical Hadamard after initializing in the X_L basis. Logical Hadamards are however somewhat complicated, as we will see later; it is actually simpler to initialize in the Z_L basis directly. This is shown in Fig. 31. Starting with a fully-stabilized array, a strip of X stabilizers is turned off, where each end of the strip becomes one of the two qubit holes. The Z stabilizers that border the strip are also switched from measuring four adjacent data qubits to measuring three qubits. In Fig. 31b, there are six Z stabilizers that are switched from four- to three-terminal operation. Note that the stabilizer measurements still all commute, as you can check. The three isolated data qubits (numbered 1, 2 and 3 in Fig. 31) are

measured once along Z to maintain error correction, and then the data qubits are set to the ground $|0\rangle$ state as in Fig. 31c. Setting these three qubits to their ground state ensures that the quiescent state $|\psi\rangle$ that results will be in the logical ground state of the $Z_L = Z_1Z_2Z_3$ operator. Finally the two X stabilizers internal to the strip are turned back on, and the three-terminal Z stabilizers switched back to four-terminal measurements, completing the process. If instead a $|1_L\rangle$ is desired, an X_L is applied (in software) to bit-flip the logical qubit.

Note that the projective measurements of the two X stabilizers that were turned back on at the end of the initialization will leave the three data qubits in a $+1$ eigenstate of Z_L, in other words a $|0_L\rangle$ eigenstate, even though the data qubits themselves will no longer be in their individual ground states $|0\rangle$. We can see this as follows: After the ground state re-set, the quiescent state $|\psi\rangle$ is transformed to $|\psi'\rangle = |000\rangle \otimes |\phi\rangle$, where $|000\rangle$ is the state of the three data qubits, and $|\phi\rangle$ the state of all the other data qubits in the array. The state $|\psi'\rangle$ is a $+1$ eigenstate of Z_L, but is not an eigenstate of all the X stabilizers, as you can easily verify. However, we know that Z_L commutes with all the X stabilizers, so there are common eigenstates of both Z_L and these stabilizers. The state $|\psi'\rangle$ can be written as a superposition of these X stabilizer eigenstates, which are all $+1$ eigenstates eigenstates of Z_L, so the logical qubit remains in $|0_L\rangle$. The X stabilizer measurements will project $|\psi'\rangle$ onto one of these eigenstates, leaving us with a state $|\psi''\rangle$ that is still a $+1$ eigenstate of Z_L but is also an eigenstate of all the X (and Z) stabilizers in the array.

Initializing a Z-cut qubit is completely analogous to the initialization of an X-cut qubit.

12.2 Measurement

Measuring a logical qubit is almost the inverse of initialization. As with initializing, logical measurements can be classified as "easy" or "difficult". For an easy measurement, the measure qubits in the qubit holes are simply turned on, with the stabilizer measurement projecting the logical qubit onto a stabilizer eigenstate, with the stabilizer eigenvalue equal to the logical qubit measurement outcome (after completing d surface code cycles and voting). For the X-cut qubit shown in Fig. 32a, we turn on the two measure-X qubits in the holes, whose measurement projects the data qubits adjacent to them into an $X_aX_bX_cX_d$ product eigenstate. This constitutes an X_L measurement of the logical qubit, with the X_L eigenvalue equal to the (time-stable) value of X_{abcd}.

For a difficult measurement, used to e.g. measure the X-cut qubit in the Z_L basis, we use the measurement process illustrated in Fig. 32.

A completely analogous process is used to perform an "easy" measurement of Z_L or a "difficult" measurement of X_L for a Z-cut qubit.

13 Moving qubits

We turn now to the very interesting method used to entangle logical qubits. This is done by physically moving the qubit holes in the 2D array, providing a central and unique functionality of the surface code. By passing one hole of a two-hole qubit between the two holes of a second qubit, "braiding" the logical qubits together, you perform a logical CNOT of the two qubits!

We first cover how to move a logical qubit hole while preserving the surface code error protection. The corresponding transformations of the logical operators associated with the moving qubit are described in the Heisenberg representation. We briefly review the Heisenberg representation of quantum mechanics (see e.g. [58] for a more complete introduction, and [59] in relation to quantum computing).

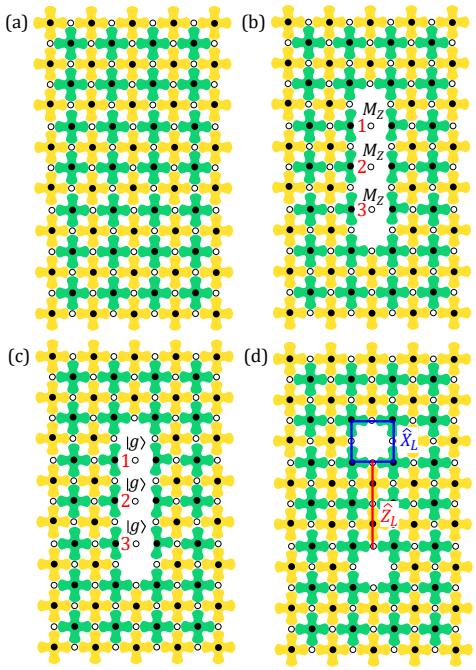

Figure 31: A "difficult" Z eigenstate initialization of an X-cut logical qubit. (a) Starting with a fully stabilized array, (b) turn off a column of four measure-X qubits, switch the adjacent measure-Z qubits from 4- to 3-terminal measurements, and perform Z measurements of the data qubits numbered 1, 2 and 3, to maintain error tracking. (c) Reset the data qubits 1, 2 and 3 to $|g\rangle$, and (d) turn two of the measure-X qubits back on, switch the adjacent measure-Z qubits back to 4-terminal measurements, resulting in a logical qubit initialized in $|g_L\rangle$ due to step (c). Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

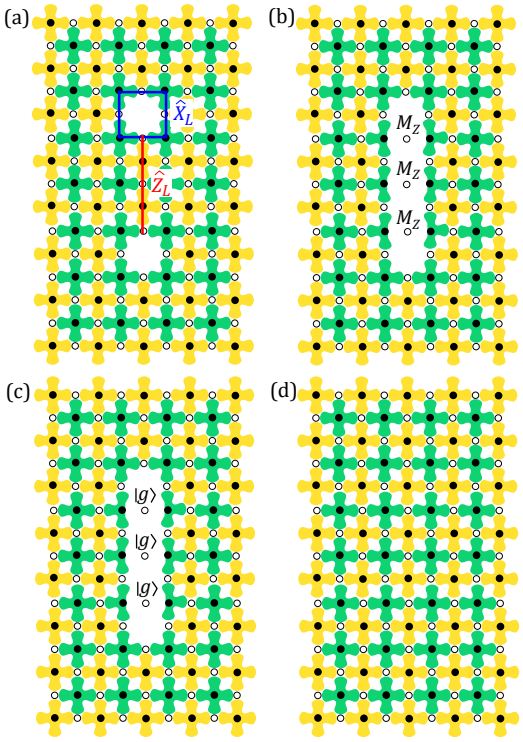

Figure 32: A Z-axis ("difficult") measurement process for an X-cut logical qubit. (a) Starting with an X-cut qubit, (b) turn off the measure-X qubits between the two qubit cuts while also switching the neighboring measure-Z qubits from four- to three-terminal measurements. Measure the un-stabilized data qubits along Z. The product of the measurement outcomes is the measurement of Z_L. (c) Reset the qubits to their ground states $|g\rangle$ and (d) destroy the logical qubit by turning on all the stabilizers. Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

Many processes comprise unitary transformations U, with $UU^\dagger = U^\dagger U = I$. These include changes of basis, changes of representation, and temporal evolution. In the Schrödinger representation, these transformations are applied to the wavefunction, so that $|\psi\rangle \Rightarrow U|\psi\rangle$, with the operators kept static. Hence for example wavefunctions are time-dependent, arising from the application of the evolution operator $U(t)$, while operators are time-independent. Inner products are invariant under a unitary transformation.

In the Schrödinger picture, the matrix elements of an operator A transform according to

$$\begin{aligned} \langle\phi|A|\psi\rangle &\Rightarrow \left(\langle\phi|U^\dagger\right) A \left(U|\psi\rangle\right) \\ &= \langle\phi| \left(U^\dagger A U\right) |\psi\rangle. \end{aligned} \tag{93}$$

As in general $A \neq U^\dagger A U$, the matrix elements can change under a unitary transformation, even though the operators do not. The second line in Eq. (93) provides the basis for the Heisenberg representation: We can equivalently assume the wavefunctions $|\psi\rangle$ do not change under the unitary transformation U, and instead we modify the operator $A \Rightarrow A' = U^\dagger A U$. Hence in the Heisenberg picture, operators rather than wavefunctions are modified under a change of basis, and operators are time-dependent. The evolution of any measurable quantity will be the same in the Heisenberg representation as in the Schrödinger representation, so there is no detectable difference between the two pictures: They are thus completely equivalent from the point of view of measurement outcomes.

In the surface code, moving and braiding qubits are transformations that affect both the physical array of qubits as well as the logical operators. The physical transformations are not unitary, as these involve projective measurements of the data qubits, but the transformations of the logical operators are unitary, and are most easily described using the Heisenberg representation.

13.1 One-cell logical qubit move

In Fig. 33 we show how to move a Z-cut qubit hole downwards by one cell in the surface code array. The logical qubit has the logical operators $X_L = X_1X_2X_3$ and $Z_L = Z_3Z_4Z_5Z_6$, where Z_L is the loop of Z operators surrounding the lower Z-cut hole.

We first describe the physical operations involved in the move, which takes two complete surface code cycles. First, we stop measuring the Z stabilizer just below the qubit hole that we are moving. This means that data qubit 6 will not be measured by a Z stabilizer (see Fig. 33). The two measure-X qubits adjacent to data qubit 6 are converted from four- to three-terminal measurements, excluding data qubit 6 in their measurement cycles. The next surface code cycle is then executed, during which data qubit 6 is measured separately with the now-idle Z stabilizer in X, yielding in its eigenvalue X_6; this measurement is used for tracking errors, and for monitoring the sign of the redefined X_L operator. After completing this cycle, the measure-Z qubit in the original lower Z-cut hole is turned on, so it begins to perform a normal measurement cycle. The two measure-X qubits that monitor data qubit 6 are also switched back to four-terminal measurements. The next surface code cycle is executed, which completes the physical operation for the move. To establish all stabilizer values in time takes an additional $d-1$ surface code cycles. Altogether this one-cell move takes $1+d$ surface code cycles.

While performing the physical operations, the two logical operators Z_L and X_L need to be redefined to preserve their functionality. We redefine Z_L so that it encloses two stabilizer cells, its original cell plus the one below it, by multiplying Z_L by the four Z operators, $Z_6Z_7Z_8Z_9$, that make up the Z stabilizer below the qubit hole. We term the

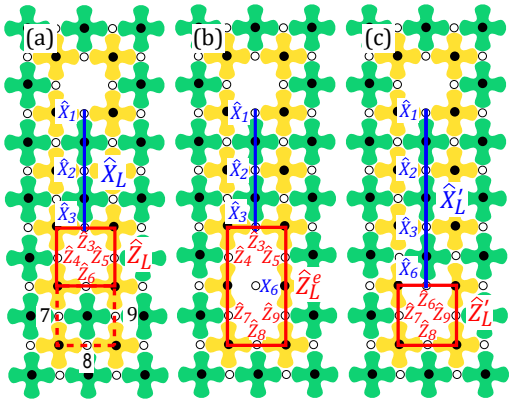

Figure 33: Moving a Z-cut logical qubit hole down by one cell. (a) Logical qubit with $X_L = X_1X_2X_3$ and $Z_L = Z_3Z_4Z_5Z_6$. (b) Z_L extended by multiplying it by the Z stabilizer just below the lower qubit hole: $Z_L^e = (Z_6Z_7Z_8Z_9)Z_L = Z_3Z_4Z_5Z_7Z_8Z_9$. This Z stabilizer is turned off along with converting the surrounding four-terminal X stabilizers into three-terminal stabilizers, leaving data qubit 6 un-stabilized. An X measurement of data qubit 6 is performed. (c) Extend operator X_L' by multiplying X_L by X_6: $X_L' = X_6X_L = X_1X_2X_3X_6$. The Z stabilizer just above qubit 6 is turned on, we wait d surface code cycles. Define Z_L' as the product of this Z stabilizer and Z_L^e: $Z_L' = Z_6Z_7Z_8Z_9$. Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

new extended operator Z_L^e, given by

$$\begin{aligned} Z_L^e &= (Z_3Z_4Z_5Z_6) \times (Z_6Z_7Z_8Z_9) \\ &= Z_3Z_4Z_5Z_7Z_8Z_9, \end{aligned} \tag{94}$$

as shown in Fig. 33a. We complete the first surface code cycle of the move, turning off the lower Z stabilizer and measuring data qubit 6. We redefine the X_L operator by multiplying it by X_6, giving

$$X_L' = X_1X_2X_3X_6; \tag{95}$$

see Fig. 33b.

The original qubit hole stabilizer is turned back on, and the second surface code cycle of the move completed. The Z_L^e operator is redefined by multiplying it by the four Z operators that make up the stabilizer that was just turned on, $Z_3Z_4Z_5Z_6$, giving

$$\begin{aligned} Z_L' &= (Z_3Z_4Z_5Z_6) \times Z_L^e \\ &= (Z_3Z_4Z_5Z_6) \times (Z_3Z_4Z_5Z_7Z_8Z_9) \\ &= Z_6Z_7Z_8Z_9. \end{aligned} \tag{96}$$

The two new logical operators X_L' and Z_L' are as drawn in Fig. 33c. Note that, as required, they still share a single data qubit and commute with all the stabilizers.

13.2 Byproduct operators

When performing a move transformation, the extension of X_L by multiplying by X_6 can yield an extended X_L' that differs in sign from X_L; this occurs if the measurement outcome $X_6 = -1$. Similarly the final Z_L' can differ in sign from Z_L, depending on the product of

the two Z stabilizer measurement outcomes involved in the move. Given the initial state $|\psi\rangle$ prior to the transformation, this results in

$$|\psi\rangle \Rightarrow X_L'^{p_Z} Z_L'^{p_X} |\psi'\rangle, \tag{97}$$

where $|\psi'\rangle$ is the desired state, which would result if there were no sign changes in the logical operators. Hence we must operate on the resulting state with the operator product $Z_L'^{p_X} X_L'^{p_Z}$ to regain the desired state $|\psi'\rangle$.

These additional bit- and phase-flip operators are called "byproduct operators". The powers p_X and p_Z are determined by whether a sign change occurred in X_L during the move. The byproduct operator Z_L' corrects the sign of the transformed X_L' through their anti-commutation relation, and the byproduct operator X_L' similarly corrects Z_L'. Note that these operators are not actually applied, but are tracked in the control software. These corrective operators are thus only applied when the logical qubit is measured, reversing the sign of the measurement if needed. Two corrective X_Ls or two corrective Z_Ls of course cancel.

We can also look at what happens to our description of the 2D array wavefunction $|\psi\rangle$ in terms of the stabilizer and logical qubit subspaces. Prior to the move, we have $|\psi\rangle = |Q\rangle|q_L\rangle$; after the move, we have $X_L'^{p_Z} Z_L'^{p_X} |\psi'\rangle = X_L'^{p_Z} Z_L'^{p_X} |Q'\rangle|q_L'\rangle$, where $|Q'\rangle|q_L'\rangle$ is the desired state. The byproduct operators only affect the logical state, so we can write the equivalent to Eq. (97) as

$$|q_L\rangle \rightarrow X_L'^{p_Z} Z_L'^{p_X} |q_L'\rangle, \tag{98}$$

where $|q_L'\rangle$ is the desired logical state.

13.3 Multi-cell logical qubit move

We can easily generalize the one-cell move to a multi-cell move, using the same number of surface code clock cycles as a one-cell move, translating the logical qubit hole over an unlimited number of cells. Multi-cell moves are performed by extending the one-cell move to a contiguous strip of cells. A multi-cell move is shown in Fig. 34 for a Z-cut qubit hole. Details for the multi-cell move are given in the figure caption.

Briefly, Z_L is extended by multiplying it by all the stabilizers through which the logical qubit is to be moved, giving the extended Z_L^e:

$$Z_L^e = (Z_{s2} Z_{s3} \ldots Z_{s,n}) Z_L. \tag{99}$$

Here each stabilizer operator Z_{sj} represents the product of the four Z operators on the four data qubits surrounding the jth stabilizer cell. These stabilizers are then turned off, and the four-terminal X stabilizers adjacent to the strip switched to three-terminal measurements. The un-stabilized data qubits in the strip are all measured along X, and X_L is extended by multiplying it by all the X operators on these data qubits,

$$X_L' = (X_1 \ldots X_{n-1}) X_L. \tag{100}$$

We complete the move for Z_L by then multiplying Z_L^e by all the stabilizers except the one that is in the new shifted qubit hole,

$$\begin{aligned} Z_L' &= (Z_{s1} Z_{s2} \ldots Z_{s,n-1}) Z_L^e \\ &= (Z_{s1} \ldots Z_{s,n-1})(Z_{s2} \ldots Z_{s,n}) Z_L \\ &= Z_{s1} Z_{s,n} Z_L \\ &= Z_{s,n} \end{aligned} \tag{101}$$

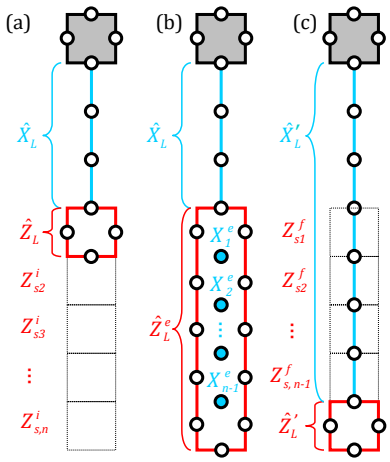

Figure 34: Moving a Z-cut qubit hole through multiple cells. The X_L chain is a blue (light) line linking the logical qubit holes and passing through the data qubits (open circles), and the Z_L loop is in red (gray). The upper (gray) and lower (white) Z-cut holes enclose idle Z stabilizers; the outlines have measure-X qubits on the vertices with data qubits on the edges. (a) Initial state, showing the logical operators X_L and Z_L, with the initial (pre-move) software-corrected Z stabilizer measurement outcomes Z^i_{s2}, $Z^i_{s3} \ldots Z^i_{s,n}$. The Z stabilizers involved in the next step are shown with thin black outlines; the jth stabilizer $Z_{s,j}$ is the stabilizer cell marked by that stabilizer's initial measurement outcome $Z^i_{s,j}$. (b) Extension of Z_L to Z^e_L, with X measurements of the isolated data qubits yielding the measurement eigenvalues X^e_1, $X^e_2, \ldots X^e_{n-1}$ (data qubits filled in blue (light)). (c) Final state, with Z'_L the shifted Z_L logical operator and X'_L the extended chain for X_L, with the Z stabilizers (thin black outlines) turned back on; after waiting d surface code cycles, we establish the final (post-move) measurement outcomes Z^f_{s1}, $Z^f_{s2}, \ldots Z^f_{s,n-1}$. Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

where we use the fact that $Z_{sj}^2 = I$ and that $Z_L = Zs1$, i.e. the original Z_L is just the set of four Z data qubit operators that define Z_{s1}. After the move, we wait an additional $d-1$ surface code cycles to establish all stabilizer values in time.

A completely analogous process is used to move X-cut logical qubit holes, exchanging the roles of the X and Z stabilizers and measurements.

We note that multi-cell moves can be done very quickly, as a very long cut can be made in just one step of the surface code cycle, and the qubit holes moved in $d+1$ surface code cycles, the same as for a one-cell move. This therefore enables long-distance interaction and communication between logical qubits, a very powerful capability.

13.4 Errors during move transformations

Previously we discussed handling of errors when the surface code stabilizers are being manipulated. A very similar discussion applies to errors that occur during the move transformations, with some differences in the details. The data qubits that are left isolated during the move can suffer from both Z and X errors; X errors have no impact, as they are erased by the single data qubit X measurements during the move. Z errors will be detected by computing the product of each data qubit X measurement with its corresponding three-terminal X stabilizer measurement, and comparing this product with that stabilizer's prior four-terminal X measurement. Z errors on the data qubits bordering the cut are detected and localized by the two X stabilizers that monitor each of these data qubits. An X error on one of these data qubits changes the sign of the solitary Z stabilizer that monitors that data qubit, but this error can be distinguished from a stabilizer measurement error by waiting d surface code cycles.

14 The braiding transformation and the logical CNOT

A "braid transformation" combines a pair of multi-cell moves involving one of the two holes of a logical qubit. The mobile hole moves along a closed path, with the first move taking the hole partway around the loop, and the second move completing the loop. As with the move transformations, braids are described in terms of their effect on the logical qubit operators. A braid transformation can entangle two logical qubits, in a way that makes the braid transformation equivalent to a logical CNOT; this is its most important function in the surface code.

14.1 Braid transformation of one logical qubit

We first describe the braid transformation of a single logical qubit, using a Z-cut qubit. In Fig. 35 we show the transformation of the X_L operator and in Fig. 36 that of the Z_L operator. For this example, the braid transformation moves the qubit hole along a path that encloses only fully stabilized cells. Later, we discuss what happens when the path encloses another logical qubit hole.

In Fig. 36, the first step shifts the qubit hole eight cells along the path (this can be any number of cells between one and the total number of cells in the braid path less the array distance d; moving the hole so that after the first move it ends up less than d cells from the start point would effectively reduce the array distance). This move is completed before the second move is started, meaning that all the stabilizers that were switched off during the move have been turned back on (except for the last cell). After waiting d surface code cycles, to catch measurement errors during the first move, the second multi-cell move is then performed, shifting the qubit hole the remaining four cells in the path, and returning

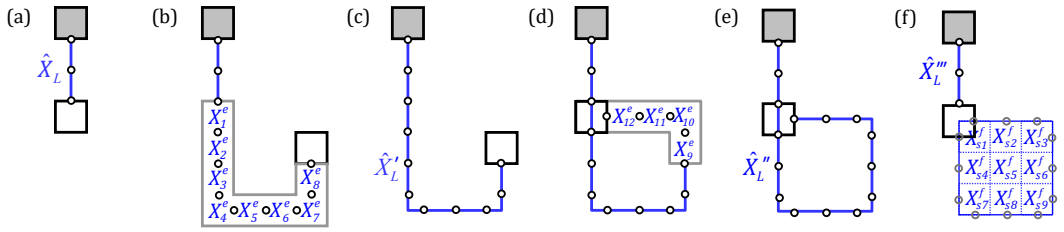

Figure 35: Braid transformation of a single Z-cut qubit. (a) Z-cut qubit with the operator X_L, showing the three data qubits that define the operator. (b) Extended opening (heavy grey) for first move in the braid. Data qubits isolated by the extension are measured in X with outcomes $X_1^e, \ldots, X_8^e$. (c) The X_L operator (blue, dark) is extended to $X_L' = X_1 \ldots X_8 X_L$. (d) The second move again involves measuring isolated data qubits in X. (e) The X_L' operator is extended to X_L'' and now includes the original chain linking the two qubit holes, and in addition a closed loop of operators. (f) The closed loop is completely stabilized by the nine X stabilizers $X_{s1} \ldots X_{s9}$ (blue (dark) squares). We can thus reduce the operator chain X_L'' to a chain X_L''' that is identical to the original X_L (other than sign difference). The signs are captured by defining the power p_X through $(-1)^{p_X} \equiv X_{s1}^f X_{s2}^f \ldots X_{s9}^f = \pm 1$, given by the product of all the X stabilizer measurements enclosed by the X_L'' loop. Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

the hole to where it started. The braid cannot be completed in a single multi-cell move, as this would isolate the part of the surface code array enclosed by the braid.

As can be seen in Figs. 35 and 36, the braid transforms the two logical operators X_L and Z_L in different ways. The X_L operator is a chain of three X operators linking the two holes (Fig. 35a). As shown in Fig. 35b-e, this operator is extended in each of the moves, such that after the second move, the X_L'' operator in Fig. 35e includes the original chain of three data qubit operators in addition to the loop $X_{\rm loop} = X_1 \ldots X_{12}$. This closed loop encloses only fully stabilized cells, with stabilizers X_{s1}, $X_{s2} \ldots X_{s9}$, so $X_{\rm loop} = X_{s1} \ldots X_{s9}$. The quiescent state $|\psi\rangle$ is necessarily an eigenstate of $X_{\rm loop}$, with eigenvalue given by the measurement outcomes of the enclosed stabilizers, $X_{\rm loop} = X_{s1} \ldots X_{s9}$. Another way to understand this is that the closed loop of operators $X_{\rm loop}$ can be deformed through all the enclosed X stabilizers, leaving only a product of measurement outcomes.

The Z_L operator is the loop of qubit Z operators that encloses the lower qubit hole. As shown in Fig. 36, the braid transformation alternately extends and then collapses the operator loop that surrounds the Z-cut hole, once for each of the two moves; other than sign changes from Z stabilizers that the loop passes over, the final Z_L loop is the same as the initial loop. The difference between the transformation of X_L and Z_L is the key to how the braid acts as a CNOT.

Details of the braid transformation involving just one logical qubit are given in the captions for Figs. 35 and 36. Note that in Figs. 35 and 36, while we separate the actions that affect the X_L operator from those that affect Z_L, all X measurements of the isolated data qubits and measurements of the Z stabilizers are performed as part of the braid.

14.2 Braiding two qubits

What if the braid path encloses another logical qubit hole? The easiest way to understand what happens is to examine how operators on the two qubits are transformed by the braid operation. The two-qubit operators will be outer products of the single-qubit operators X_L, Z_L and I_L.

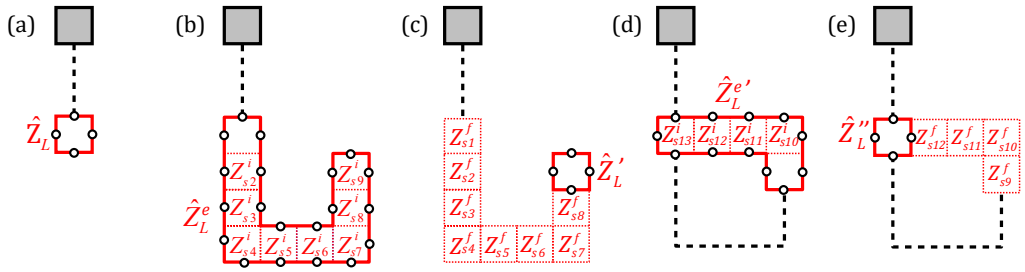

Figure 36: Braid transformation of a Z_L operator on a single Z-cut qubit. (a) Double Z-cut qubit with Z_L in red; data qubits are open circles; dashed line linking the two qubit holes does not represent an operator. (b) Extended opening (solid red loop) for first move in braid. The pre-move Z stabilizer outcomes $Z^i_{s2}, \ldots, Z^i_{s9}$ are all equal to ± 1. The Z^e_L operator (solid red loop) is extended from Z_L, $Z^e_L \equiv Z_{s2} \ldots Z_{s9} Z_L$. (c) The extended opening is closed up by turning on the stabilizers $Z_{s1}, \ldots, Z_{s8}$, and waiting d surface code cycles to obtain the error-corrected outcomes $Z^e_{s1}, \ldots, Z^e_{s8}$. The new Z'_L operator is the red (gray) loop formed by four data qubit Z operators, equal to $Z'_L = (Z_{s1} \ldots Z_{s8})(Z_{s2} \ldots Z_{s9}) Z_L$. (d) Second move in the braid, which involves extending Z'_L to $Z^{e'}_L$, using the associated Z stabilizer outcomes. (e) The final Z''_L is a closed loop of four data qubit operators identical to the original Z_L, although Z''_L may differ in sign with respect to Z_L, captured by defining $(-1)^{pz} \equiv (Z^f_{s9} \ldots Z^f_{s12})(Z^i_{s10} \ldots Z^i_{s13})(Z^f_{s1} \ldots Z^f_{s8})(Z^i_{s2} \ldots Z^i_{s9}) = \pm 1$. Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

As a review of two-qubit operators, consider $X_L \otimes Z_L$, operating on the two-qubit logical state $|a_L b_L\rangle$. Using the eigenstates $|0_L\rangle$ and $|1_L\rangle$, then $X_L \otimes Z_L$ can be represented in the standard two-qubit basis $\{|0_L 0_L\rangle, |0_L 1_L\rangle, |1_L 0_L\rangle, |1_L 1_L\rangle\}$ by the matrix

$$\begin{aligned} X_L \otimes Z_L &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}. \end{aligned} \tag{102}$$

The braid transforms each two-qubit operator into some other two-qubit operator, with the outcome depending on the specific operators, and on the order in which they appear (so $X_L \otimes Z_L$ does not transform the same way as $Z_L \otimes X_L$). The outcome of a braid depends on what types of qubits we are braiding; we focus here on braiding a Z-cut qubit through an X-cut qubit, as shown in Fig. 37. As we will see, only a braid between a Z-cut and an X-cut qubit yields an operator transformation that is equivalent to a logical CNOT. It is not possible to obtain the desired CNOT transformations when braiding two Z-cut qubits together, or two X-cut qubits; we discuss these situations later.

In Fig. 37, the Z-cut qubit is in the upper part of the figure, with the braid moving the lower Z-cut hole of this qubit around a closed loop. The X-cut qubit is in the lower part of the figure, with the braid taking the Z-cut hole between the two X-cut holes, enclosing the upper X-cut hole. As we show later, demonstrating that a braid is equivalent to a CNOT only involves showing that four of the sixteen possible two-qubit operator combinations transform correctly; these are $X_L \otimes I_L$, $I_L \otimes X_L$, $Z_L \otimes I_L$ and $I_L \otimes Z_L$. The transformations for all the other two-qubit combinations of X_L, Z_L and I_L can be constructed from these four. Here we give a brief outline of these transformations.

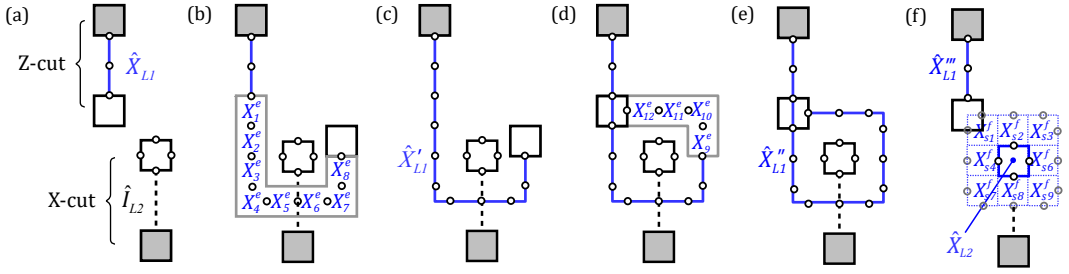

Figure 37: A braid on a Z-cut qubit, where the lower Z-cut hole is passed around the upper X-cut hole of an X-cut qubit. The braid's effect on X_{L1} operator of the Z-cut qubit 1 is shown. (a) Z-cut qubit (above) and X-cut qubit (below) with corresponding logical operators; data qubits are shown as open circles. (b) Extension for first move in braid, where data qubits are measured along X, with measurement outcomes X^e_1, $X^e_2, \ldots X^e_8$. (c) X_{L1} operator is extended in length to X'_{L1}. (d) Extension for second move in braid, where data qubits are measured along X with measurement outcomes $X^e_9, \ldots X^e_{12}$. (e) X''_{L1} operator is extended in length, comprising the original chain plus a closed loop of data qubit operators, the loop enclosing the upper hole of the X-cut qubit. (f) The loop part of the X''_{L1} operator is moved through the enclosed stabilized cells, leaving a loop of X data qubit operators that is equal to X_{L2} on the second qubit, with X_{L1} unchanged from before the braid (other than possible sign changes). Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

$X_{L1} \otimes I_{L2} \Rightarrow X_{L1} \otimes X_{L2}$: Figure 37 shows the transformation of X_{L1} on the first, Z-cut qubit, with no operation on the second qubit. As with the empty loop braid, the extension of the X_{L1} operator chain in the two moves creates an operator X''_{L1} that comprises a closed loop of X operators in addition to the original operator chain X_{L1}. The loop encloses the X-cut qubit's upper hole. For the empty loop braid, we could move the loop through all the enclosed cells, as these were all stabilized, so the loop operator resolves to a simple product of measurement outcomes. Here, we cannot do this, as the X-cut hole is inside the loop and is not stabilized; instead, we can transform the loop through each stabilized cell, until it wraps tightly around the X-cut hole. This loop of X operators is then equivalent to X_{L2}, a logical X operation on the second qubit. The remaining chain of operators in X'''_{L1} is the same as the original X_{L1} chain prior to the braid. We see therefore that the braid takes

$$X_{L1} \otimes I_{L2} \Rightarrow X_{L1} \otimes X_{L2}. \tag{103}$$

$I_{L1} \otimes X_{L2} \Rightarrow I_{L1} \otimes X_{L2}$: Figure 38 shows this braid transformation. Braiding the Z-cut qubit hole through the X-cut qubit does not generate any chains of operators from either qubit that wrap around or otherwise interact with the other qubit, so the braid transformation in this case does nothing (other than sign changes). Hence we find

$$I_{L1} \otimes X_{L2} \Rightarrow I_{L1} \otimes X_{L2}. \tag{104}$$

$I_{L1} \otimes Z_{L2} \Rightarrow Z_{L1} \otimes Z_{L2}$: Figure 39 shows this transformation. In Fig. 39b, Z_{L2} is extended as shown in the figure, by multiplying it by the Z stabilizers outlined by the dashed boxes. The first qubit's hole is then moved through the path left open by the extension of Z_{L2} (Fig. 39c). Z_{L2} is then multiplied by all the stabilizers shown in the dashed boxes (Fig. 39d), leaving behind a loop of Z operators surrounding the first qubit's hole, a loop that is exactly a Z_{L1} operation on the first Z-cut qubit, as well as the original

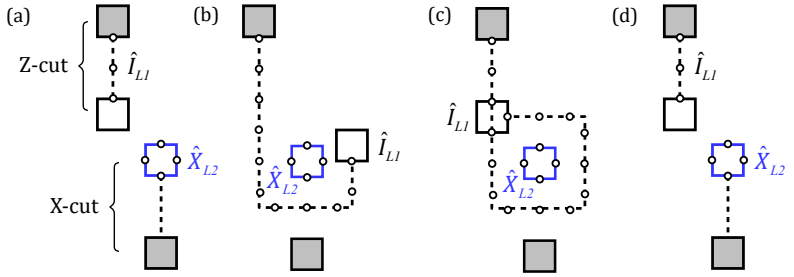

Figure 38: (a)-(d) Illustration of the braid transformation of $I_{L1} \otimes X_{L2}$. After the move, there is no net change of either operator. Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

Z_{L2} on qubit 2, unchanged from prior to the braid (other than possible sign changes). The braid therefore performs the transformation

$$I_{L1} \otimes Z_{L2} \Rightarrow Z_{L1} \otimes Z_{L2}. \tag{105}$$

$Z_{L1} \otimes I_{L2} \Rightarrow Z_{L1} \otimes I_{L2}$: This transformation involves braiding the first qubit hole through the second qubit, which drags the loop of Z operators along as it did for the empty loop braid, but as the loop preserves its closed form during the two moves, it does not generate a chain or loop of operators that can act on the second qubit, so the braid transformation does nothing (other than sign changes). Hence we find

$$Z_{L1} \otimes I_{L2} \Rightarrow Z_{L1} \otimes I_{L2}. \tag{106}$$

In general, a braid transformation leaves logical operators that are built from closed loops of data qubit operators unchanged, and there is no braid-induced interaction with the other qubit hole. By contrast, logical operators that are built from open chains of data qubit operators linking the two qubit holes end up leaving a loop of operators surrounding the other qubit hole. The different braid outcomes arise because the first logical qubit is a Z-cut qubit, for which the X_L operator is an open chain that interacts with the second qubit, while the second logical qubit is an X-cut qubit, for which the Z_L operator is an open chain that interacts with the first qubit.

The braid is made of two move transformations that induce sign changes in the first qubit's logical operators, which appear as byproduct logical operators X_{L1} and Z_{L1} acting on the array wavefunction, as was discussed for the one-cell and multi-cell moves. The braid transformation also generates sign changes in the second qubit's logical operators, even though that qubit is not displaced during the braid. These sign changes generate the byproduct logical operators X_{L2} and Z_{L2} acting on the array wavefunction.

We now turn to a discussion of the CNOT gate, and make clear why the transformations we have detailed actually identify the braid as a CNOT between two logical qubits.

14.3 The CNOT gate

The CNOT gate is a fundamental gate for quantum computation. One of the two qubits in the CNOT serves as the control, and the other as the target. In the standard two-qubit

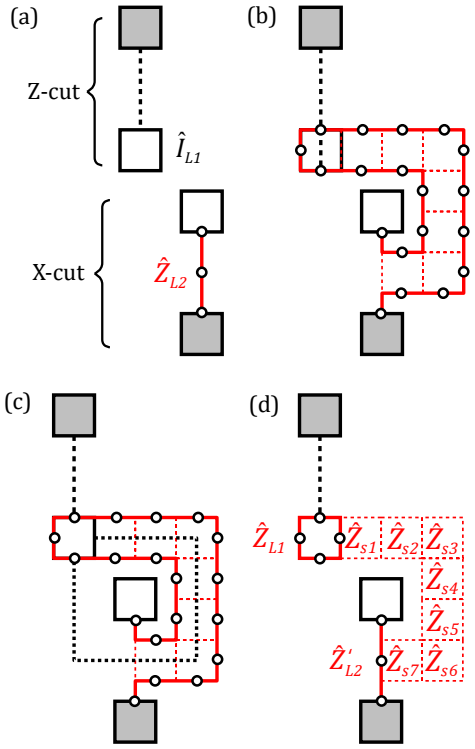

Figure 39: Braid transformation of $I_{L1} \otimes Z_{L2}$. (a) Prior to the move, displaying I_{L1} on qubit 1 and Z_{L2} on qubit 2. Data qubits are shown as open circles where relevant. (b) The Z_{L2} operator is extended by multiplying it by a set of stabilizers (dotted boxes), resulting in the heavy red line that is equivalent to the original Z_{L2}, with the exception of possible sign changes. (c) Qubit 1's hole is moved through the path opened up by extending Z_{L2}. (d) Z_{L2} is moved back to its original form, leaving behind a loop of Z physical qubit operators that comprise a Z_{L1} operator on qubit 1. Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

basis $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$, the CNOT is represented by the 4×4 real matrix

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{107}$$

If the control qubit is in $|0\rangle$, the target qubit state is unchanged, while if the control qubit is in $|1\rangle$, the target qubit undergoes an X bit-flip, exchanging $|0\rangle$ and $|1\rangle$. Note that C is Hermitian, $C^\dagger = C$, and unitary, $CC^\dagger = C^\dagger C = CC = I$.

One way to test a potential CNOT operation is to perform the operation on each of the four two-qubit basis states, and then do projective measurements of the result onto each of the four basis states. These sixteen experiments can be compared to the matrix in Eq. (107) to verify that the CNOT has been implemented correctly. This is essentially a Schrödinger picture test of the CNOT.

An equivalent method is to use the Heisenberg picture, by examining the transformation of the various two-qubit operators due to the action of the CNOT. This would seem to involve showing that the CNOT performs the correct transformation for all sixteen outer products of pairs of the four single qubit operators, I, X, Y and Z. It turns out you only need to show that the CNOT transforms four of these outer products correctly:

$$\begin{aligned} C^\dagger (I \otimes X) C &= I \otimes X, && (108) \\ C^\dagger (X \otimes I) C &= X \otimes X, && (109) \\ C^\dagger (I \otimes Z) C &= Z \otimes Z, \text{ and} && (110) \\ C^\dagger (Z \otimes I) C &= Z \otimes I. && (111) \end{aligned}$$

The other twelve relations are either trivial ($I \otimes I$ is unchanged), or can be written in terms of these four transformations. Hence $C^\dagger (X \otimes X) C = X \otimes I$ is the same as Eq. (109), as can be seen by multiplying both sides of Eq. (109) by $C^\dagger$ on the left and by C on the right, and using the fact that C is unitary. Combinations such as $X \otimes Z$ transform according to

$$\begin{aligned} C^\dagger (X \otimes Z) C &= C^\dagger (X \otimes I) (I \otimes Z) C \\ &= C^\dagger (X \otimes I) C C^\dagger (I \otimes Z) C \\ &= (X \otimes X) (Z \otimes Z) \\ &= XZ \otimes XZ \\ &= Y \otimes Y. \end{aligned} \tag{112}$$

Other combinations involving Y can be worked out using the identity $Y = ZX$; hence

$$\begin{aligned} C^\dagger (Y \otimes I) C &= C^\dagger (ZX \otimes I) C \\ &= C^\dagger (Z \otimes I) (X \otimes I) C \\ &= C^\dagger (Z \otimes I) C C^\dagger (X \otimes I) C \\ &= (Z \otimes I) (X \otimes X) \\ &= ZX \otimes IX \\ &= Y \otimes X. \end{aligned} \tag{113}$$

We can therefore validate a CNOT implementation by verifying that it satisfies the four relations given by Eqs. (108-111). However, these are precisely the four transformations that we worked out for the braid, so indeed a braid is a CNOT.

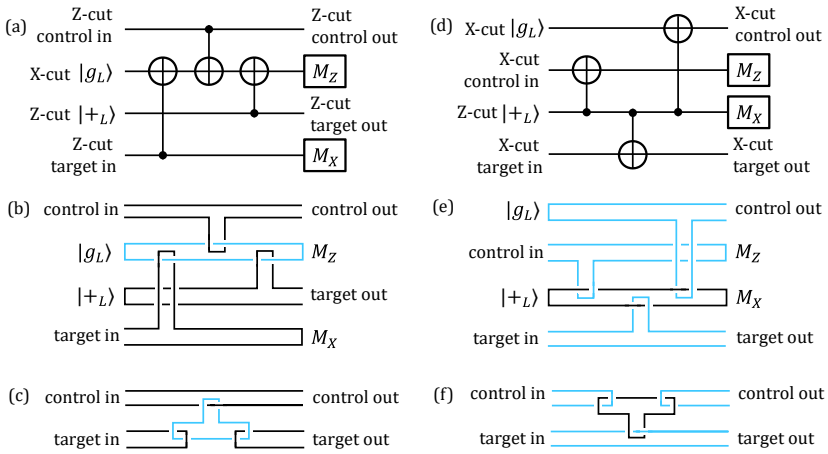

Figure 40: (a) Logical CNOT between two Z-cut qubits. The control, target in, and the second ancilla (in $|+_L\rangle$) are all Z-cut qubits, while the first ancilla is an X-cut qubit. The logical CNOTs are all between Z- and X-cut qubits, generated by braiding transformations. The measure outcomes M_Z and M_X signal how the output state should be interpreted, as described in the main text. (b) Simplified representation using braids, with black lines for Z-cut qubits and blue (light) lines for the X-cut qubit. There are a pair of lines for each logical qubit, one line for each qubit hole. The two lines join when a logical qubit is created or measured. (c) Even more condensed representation. (d) Logical CNOT between two X-cut qubits. The control, target in, and the first ancilla are all X-cut qubits, while the second ancilla is a Z-cut qubit. The logical CNOTs are all between X- and Z-cut qubits, generated by braiding transformations with the Z-cut as the control. The measure outcomes M_Z and M_X signal how the output state should be interpreted. (e) Simplified representation, with black lines for the Z-cut qubits and blue (light) for the X-cut qubit. (f) Even more simplified representation. Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

Note that the full braid transformation, including all the operations on the physical data qubits, involves a number of projective measurements and is therefore not unitary. However, if we consider the effect of the braid on the product $|Q\rangle|q_L\rangle$, while the transformation of the stabilized state $|Q\rangle$ is not unitary, the transformation of the logical state $|q_L\rangle$ is indeed a unitary one.

14.4 CNOT between two Z-cut qubits

We have only shown how to perform CNOTs using a braid with a Z-cut qubit as the control and an X-cut qubit as the target. We can extend this to a CNOT between two Z-cut qubits using the circuit shown in Fig. 40a-c. The circuit performs a CNOT of the logical Z-cut "target-in" qubit using the logical Z-cut control qubit, with the Z-cut "target-out" qubit carrying the result. An ancillary X-cut qubit is the target for the three logical CNOTs in the circuit, so all the CNOTs are braid transformations between a Z-cut and an X-cut qubit. The circuit includes two logical measurements, with outcomes M_Z and M_X. If the target-in qubit is measured to be in $|+_L\rangle$ ($|-_L\rangle$), with $M_X = +1(-1)$, then the target-out qubit does not (does) have a Z_L applied to it prior to the CNOT. If the X-cut qubit is measured to be in $|g_L\rangle$ ($|e_L\rangle$), with $M_Z = +1(-1)$, then the target does not (does) have an X_L applied to it after the CNOT.

You can verify that this circuit works properly by using operator transformations, or alternatively by testing it with the four input basis states $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$ for the control and target-in Z-cut qubits (dropping the logical subscript L for now). There is an analogous circuit for performing a CNOT between two X-cut qubits, using a Z-cut qubit as an intermediate ancillary to perform the braid.

Problem 17: Verify that the circuit in Fig. 40(a)-(c) works as promised.

14.5 Single-control, multi-target CNOTs

It is frequently necessary to implement single-control, multi-target CNOTs; these appear for example in the distillation circuits used to purify imperfect states, as we shall see when we discuss the S and T gates. It turns out that these kinds of CNOTs are actually no more complicated than a single-control, single-target CNOT, and in fact can be implemented in the same number of surface code cycles as a CNOT between two Z-cut or between two X-cut qubits. We don't show the details here, but refer the reader to the literature, such as Ref. [28].

15 The Hadamard transformation

The Hadamard transformation is a single-qubit gate that in the standard qubit $|0\rangle$, $|1\rangle$ basis is represented by the 2×2 matrix

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{114}$$

In the Heisenberg representation, the Hadamard takes X to Z and vice versa, i.e.

$$\begin{aligned} H^\dagger X H &= Z, \\ H^\dagger Z H &= X. \end{aligned} \tag{115}$$

Problem 18: Show the relations in Eq. (115) are correct.

A logical Hadamard is implemented in the surface code as shown in Figs. 41, 42 and 43; see Ref. [60]. We start in Fig. 41 with an array of $d = 7$ logical qubits. Next we isolate the logical qubit in a separate patch of the 2D array, shown in Fig. 42a. The Z_L operator loop is transformed to a patch-crossing chain by multiplying it by a number of stabilizers in the patch. By widening the ring that isolates the logical qubit into a "moat", so that the moat engulfs the two qubit holes, the logical qubit is transformed to a simple "patch" qubit as shown in Fig. 42b-d, similar to the $d = 5$ array qubit we discussed earlier.

The key to the logical Hadamard is now implemented, by performing *physical* Hadamards on all the data qubits in the patch; this exchanges the X_L operator for Z_L and vice versa, as well as swapping the identities of the X and Z stabilizers (Fig. 42e). This however results in a misalignment of the stabilizers in the patch with those in the larger 2D array, so we then perform two swaps, from data qubit to measure qubit, then from measure qubit to data qubit, shifting the patch by one stabilizer cell and realigning the stabilizers (Fig. 42f). The two Z-cut holes are then recreated (Fig. 43g), positioned so that the Hadamard-transformed X_L chain ends on the internal boundary of each hole, and the Z_L chain is multiplied by a set of stabilizers that returns it to a loop around one of the

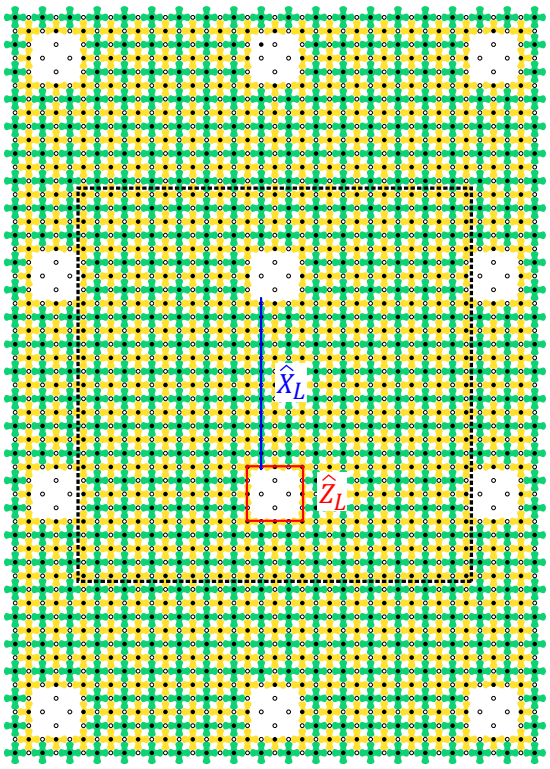

Figure 41: Executing the logical Hadamard. The spacing and size of the Z-cut qubit holes corresponds to a distance $d = 7$. The two holes in the center of the array form the logical qubit that is the target for the logical Hadamard, and for which we display the X_L and Z_L logical operators. The dashed box outlines the limits for what is shown in Figs. 42 and 43. Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

qubit holes. The qubit holes are then moved to realign them with their original positions (Fig. 43h-j), a move that is split into two steps to preserve the array distance d. In the final step, the qubits are rejoined with the main array (as in Fig. 41).

16 Single qubit S_L and T_L operators

Finally, we need to implement the S_L and T_L operators and their adjoints. These operators are represented in the standard basis by the matrices

$$S_L = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \tag{116}$$

and

$$T_L = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}. \tag{117}$$

The S_L gate is also called the P or phase gate, and T_L the $\pi/8$ gate.

As we will see, the circuit used to implement T_L is probabilistic, and half the time generates $T_L^\dagger$. As you can easily check using Eq. (116), when this occurs $T_L^\dagger$ can be converted to T_L by application of an S_L gate. T_L can be similarly converted to $T_L^\dagger$ by application of S_L, where $S_L T_L|\psi_L\rangle = Z_L T_L^\dagger|\psi_L\rangle$, and the Z_L byproduct operator is handled in software.

A high-fidelity logical implementation of these two gates involves special ancilla states. Implementing S_L relies on the $|Y_L\rangle$ ancilla state

$$|Y_L\rangle = \frac{1}{\sqrt{2}}\left(|g_L\rangle + i|e_L\rangle\right), \tag{118}$$

while implementing T_L relies on the $|A_L\rangle$ ancilla state

$$|A_L\rangle = \frac{1}{\sqrt{2}}\left(|g_L\rangle + e^{i\pi/4}|e_L\rangle\right). \tag{119}$$

The $|Y_L\rangle$ and $|A_L\rangle$ ancilla states are created in a special "short qubit," which can be put in an arbitrary but concomitantly imperfect state, a process known as "state injection." Once the state has been injected, the "short qubit" is increased to the standard distance d to make it less error-prone, and the imperfect logical state of the standard-distance qubit is then purified by a high-fidelity process known as "magic distillation", a process that can be found in the literature. The S_L and T_L gates are then implemented with circuits using logical CNOTs and Hadamards involving these ancilla states, as shown in Fig. 44 and Fig. 45, respectively.

The S_L gate implementation shown in Fig. 44 involves two logical CNOTs and two logical Hadamard operations. An input state $|\psi_L\rangle$ in one logical qubit is deterministically transformed into $S_L|\psi_L\rangle$ by interacting with the ancilla qubit in the $|Y_L\rangle$ state. This can be most easily seen by testing the circuit with the input state $|\psi_L\rangle = \alpha|g\rangle + \beta|e\rangle$.

To instead apply $S_L^\dagger$, we use the identity $S_L^\dagger = Z_L S_L$, which means we use the circuit shown in Fig. 44 and have a byproduct Z_L appear on the output, in other words $|\psi_L\rangle$ transforms to $Z_L(S_L^\dagger|\psi_L\rangle)$.

Problem 19: Show that the circuit in Fig. 44 performs an S operation as claimed.

The T_L gate is implemented with the non-deterministic circuit shown in Fig. 45. Given the input ancilla state $|\theta_L\rangle = |g_L\rangle + e^{i\theta}|e_L\rangle$, the output $|\phi_L\rangle$ of this circuit is

$$|\phi_L\rangle = X_L^{p_Z} R_Z\left((-1)^{p_Z}\theta\right)|\psi_L\rangle. \tag{120}$$

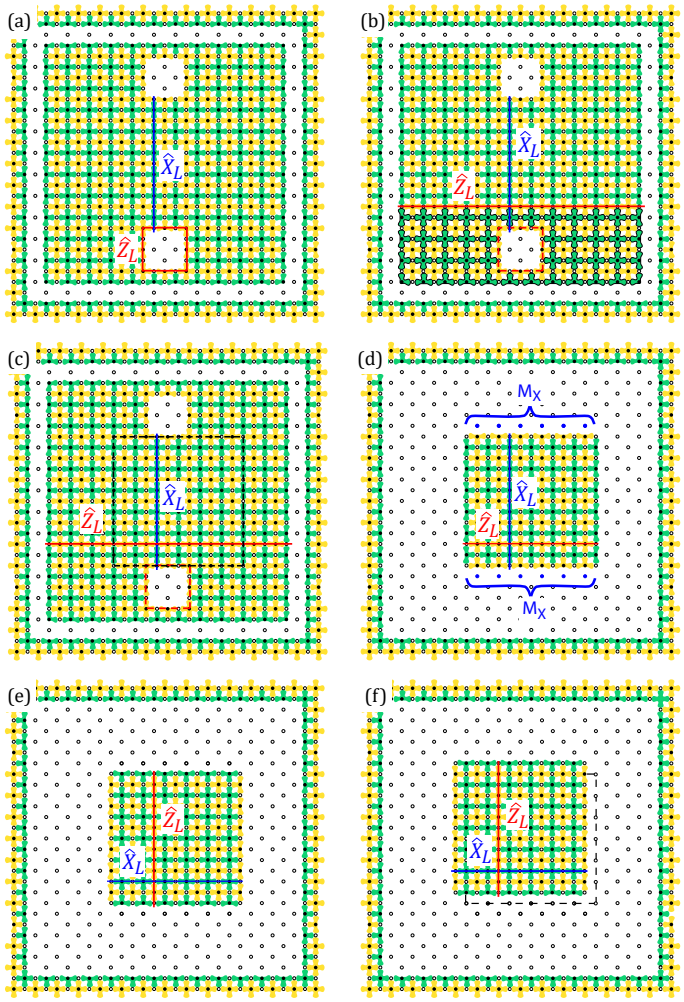

Figure 42: (a) In the first cycle, a ring of X stabilizers surrounding the target qubit is turned off, and the bordering Z stabilizers reduced to three- and two-terminal measurements. The data qubits within the ring are measured in the Z basis to maintain error tracking. This measurement pattern is applied three times (see Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a)). (b) The Z_L operator is multiplied by all the black outline Z stabilizers, transforming Z_L to the chain going from left to right. (c) In the next cycle, all X and Z stabilizers outside the dashed box are turned off, creating a "moat" as in (d), eliminating the two qubit holes. In the same cycle, all isolated data qubits are measured in Z, except those colored blue (light), which are measured in the X basis to preserve error tracking. The other un-stabilized data qubits do not need to be measured, as changes in their quantum states will be accounted for later. (e) Before the next surface code cycle starts, a physical Hadamard is performed on all data qubits, swapping X_L and Z_L. (f) Following the Hadamard, a pair of swaps is performed between each patch data qubit and its neighboring measure qubits, first between each data qubit and the measure qubit above it, then between each measure qubit and the data qubit to its left. This shifts the patch by one stabilizer cell in the array, aligning the measure qubits. The dashed box shows the location of the patch prior to these swaps. This sequence continues in Fig. 43. Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

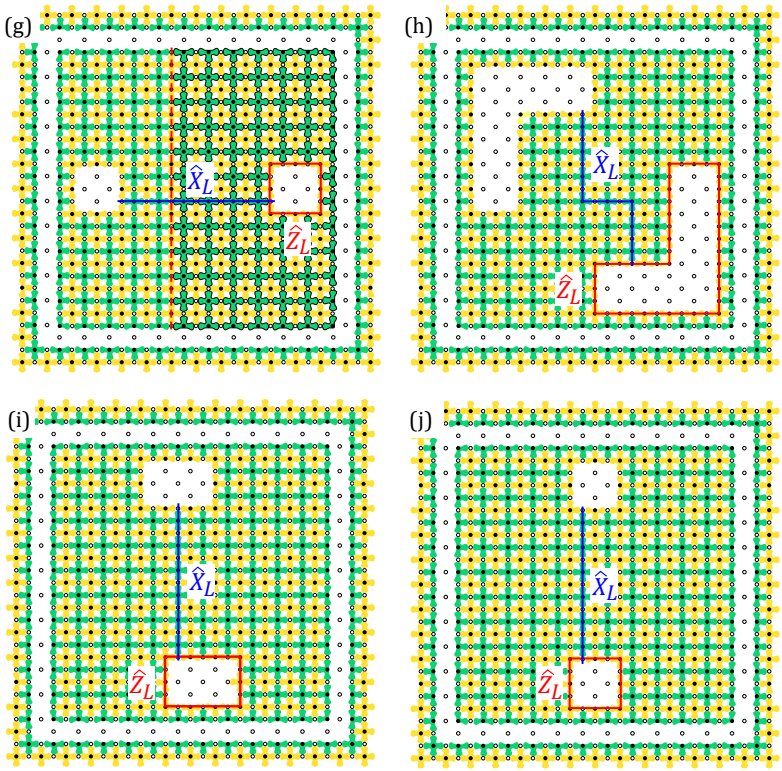

Figure 43: (continued from Fig. 42) (g) In the next surface code cycle, most of the X and Z stabilizers are turned back on, leaving two Z-cut enlarged holes and a ring, one stabilizer cell wide, isolating the 2D patch from the array, as in (a). The Z-cut holes are positioned so that the X_L chain ends on the internal boundary of each hole. The Z_L chain (dashed red line) is multiplied by all the black outline Z stabilizers, leaving a Z_L loop (solid red line) that encloses the right Z-cut hole. (h) In the subsequent surface code cycle, and the first step of returning the Z-cut holes to original locations, the Z-cut holes are expanded and Z_L and X_L modified accordingly. This move is split in two steps to preserve the distance d during this process; the process pauses here for d surface code cycles to establish all values in time. (i) In the second step of returning the Z-cut holes to their original locations, the Z-cuts are expanded to encompass the original positions. The stabilizer measurements are performed twice. (j) In the final step, the cuts are reduced to their original size, as in Fig. 41. This stabilizer measurement pattern is applied three times (see Ref. Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a)). Following this, the isolated patch is reconnected by turning on the appropriate stabilizers. Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

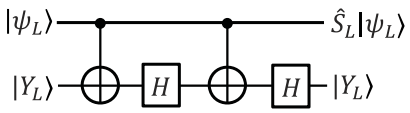

Figure 44: Logic circuit that implements the S_L gate. The circuit uses the ancilla state $|Y_L\rangle = (|g_L\rangle + i|e_L\rangle)/\sqrt{2}$, on which two controlled logical CNOTs and two logical Hadamards are performed, resulting in the input state $|\psi_L\rangle$ being transformed to $S_L|\psi_L\rangle$. Note that as $S_L^\dagger = Z_L S_L$, the same circuit transforms $|\psi_L\rangle$ to $Z_L S_L^\dagger|\psi_L\rangle$ where Z_L is a byproduct operator. Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

The first operator is a byproduct operator, whose power p_Z is equal to 0(1) if the Z_L measurement M_Z of the logical qubit state is $+1(-1)$. The second operator is a rotation by the angle θ,

$$R_Z(\theta) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}. \tag{121}$$

For the T_L gate, the rotation angle is $\theta = +\pi/4(-\pi/4)$ depending on the sign $+1(-1)$ of the Z_L measurement.

Approximately half of the times we run the circuit we will succeed in generating $T_L|\psi_L\rangle$, heralded by the measurement $M_Z = +1$. However, the other times we run the circuit the measurement $M_Z = -1$ will herald the output state

$$|\phi_L\rangle = X_L R_Z(-\pi/4)|\psi_L\rangle = X_L T_L^\dagger|\psi_L\rangle. \tag{122}$$

In this case we fix the output using $S_L T_L^\dagger = T_L$. There is a slight complication if there are byproduct X_L or Z_L operators; here we assume there are no byproduct operators. The correction to the output is achieved by applying S_L to $|\phi_L\rangle$:

$$\begin{aligned} S_L|\phi_L\rangle &= S_L(X_L T_L^\dagger|\psi_L\rangle) \\ &= X_L S_L^\dagger T_L^\dagger|\psi_L\rangle \\ &= X_L(Z_L S_L)T_L^\dagger|\psi_L\rangle \\ &= (X_L Z_L)T_L|\psi_L\rangle, \end{aligned} \tag{123}$$

where we use the identity $S_L X_L = i\, X_L S_L^\dagger$ and drop the unimportant phase factor of i. Hence passing the output state $|\phi_L\rangle$ through the S_L circuit gives the result $(X_L Z_L)T_L|\psi_L\rangle$, where $X_L Z_L$ are byproduct operators that are handled by the control software.

If we need to perform $T_L^\dagger$, we use the circuit in Fig. 45, and about half the time we will get the measurement outcome $M_Z = -1$, which indicates the circuit produced $X_L T_L^\dagger$, which is the desired output (with a byproduct operator X_L). The other half of the time we get the measurement outcome $M_Z = +1$, and we can correct the circuit output $T_L|\psi_L\rangle$ by applying S_L, as $S_L T_L = Z_L T_L^\dagger$; this is done using the S_L circuit, with the output including the byproduct operator Z_L that is handled by the control software.

Problem 20: Show that the T circuit in Fig. 45 works as advertised.

We leave some important topics for the interested reader to learn about in the literature, including the methods in the surface code for generating high-precision $|A_L\rangle$ and $|Y_L\rangle$ states using magic distillation circuits, to the level of precision as needed for whatever problem is at hand.

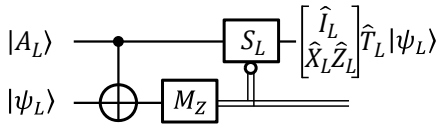

Figure 45: Logic circuit for the T_L gate. The circuit uses the ancilla state $|A_L\rangle = (|g_L\rangle + e^{i\pi/4}|e_L\rangle)/\sqrt{2}$, which is used to control a CNOT on the target state $|\psi_L\rangle$, transforming the ancilla to the output state $|\phi_L\rangle$. The CNOT target state is measured along Z_L, resulting in a projective, probabilistic outcome. Following this, the CNOT control qubit is processed by a conditional S_L gate: If the Z_L measurement M_Z yields a $+1$ outcome, the output state is the desired one, $|\phi_L\rangle = T_L|\psi_L\rangle$, and the S_L gate is not applied. If however the measurement yields a -1 outcome, the output is $|\phi_L\rangle = X_L T_L^\dagger$ and the S_L gate is applied, resulting in the output $X_L Z_L T_L|\psi_L\rangle$. Note the double lines represent the classical measurement data with a probabilistic outcome, with these data controlling the S_L gate. The bracketed operators in the output correspond to $M_Z = +1$ (I_L) and $M_Z = -1$ ($X_L Z_L$). Figure from Ref. (Fowler, Mariantoni, Martinis and Cleland, 2012a).

17 Conclusion

We have now covered all the basic aspects of the surface code approach to quantum computing. We have described all the gates that are required to implement e.g. Shor's algorithm or Grover's search algorithm. The discussion has been mostly theoretical, while the motivation for developing the surface code is of course to find a realistic and practical *physical* implementation for a quantum computer. There are a number of physical systems in which this scheme could in principle be implemented, ranging from cold atoms [61, 62] and ions [63–65], to semiconductor-based approaches [66], to superconducting integrated circuits [67–72]. Each of these systems has certain advantages and certain disadvantages. For any system to be a candidate for a surface code implementation, it must of course meet the requirements for single-qubit and two-qubit gate and measurement fidelities, which is not true for any system to date, although a number of systems are close to these requirements. The surface code clearly also requires a very large number of physical qubits (of order 10^8 is probably the smallest number needed for a practical factoring computer), so a separate requirement is the ability to assemble and integrate a large number of nominally identical qubits. Furthermore, the operation and error detection of the surface code assumes classical logic support, with the classical logic operating significantly faster than the qubits, in order that state preparation, qubit interactions, and error tracking can be maintained with a high level of fidelity.

An outstanding challenge for experimentalists, at this point, is to show the ability to assemble large (of order 100) numbers of high-fidelity qubits, and then to demonstrate that one or two cycles of error detection of the type used in the surface code can actually be implemented in a way that improves the effective qubit lifetimes. If this can be achieved, then one can begin to consider scaling up; it may be possible, using of order 1000 qubits, to demonstrate significant extensions of the lifetime of the logical qubits defined in this array compared to the lifetime of the underlying physical qubits. The lifetime of the logical qubits can then hopefully be demonstrated to scale as predicted for the distance d of the surface code.

This would then set the stage for expanding to larger arrays, demonstrating the hoped-for exponential improvement in qubit lifetimes with the size of the logical qubits. One can then begin braiding operations, and demonstrate the implementation of a logical CNOT

with the improved lifetimes due to the surface code. These goals can be met, assuming fidelities can be maintained, with of order a few thousand qubits.

Scaling up then to a full quantum computer would be mostly an engineering challenge, of how to make and then operate larger numbers (millions) of qubits. While clearly an incredibly daunting challenge, successfully meeting this challenge would make a fundamental demonstration of a complex engineered quantum system, and might usher in an era of a new computing paradigm.

18 References

References

[1] A. Y. Kitaev, *Quantum error correction with imperfect gates*, In O. Hirota, A. S. Holevo and C. M. Caves, eds., *Quantum communication, computing and measurement*, pp. 181–188. Plenum Publishing Corporation (1997).

[2] A. Y. Kitaev, *Quantum computations: algorithms and error correction*, Russian Math Surveys **52**, 1191 (1997).

[3] A. Y. Kitaev, *Fault-tolerant quantum computation by anyons* (1997), `quant-ph/9707021`.

[4] A. Y. Kitaev, *Fault-tolerant quantum computation by anyons*, Annals of Physics **303**, 2 (2003).

[5] S. B. Bravyi and A. Y. Kitaev, *Quantum codes on a lattice with boundary* (1998), `quant-ph/9811052`.

[6] M. H. Freedman and D. A. Meyer, *Projective plane and planar quantum codes*, Foundations of Computational Mathematics **1**, 325 (2001).

[7] E. Dennis, A. Y. Kitaev, A. Landahl and J. Preskill, *Topological quantum memory*, J. Math. Phys. **43**, 4452 (2002).

[8] C. Wang, J. Harrington and J. Preskill, *Confinement-higgs transition in a disordered gauge theory and the accuracy threshold for quantum memory*, Annals of Physics **303**, 31 (2003).

[9] R. Raussendorf, J. Harrington and K. Goyal, *A fault-tolerant one-way quantum computer*, Annals of Physics **321**, 2242 (2006).

[10] R. Raussendorf and J. Harrington, *Fault-tolerant quantum computation with high threshold in two dimensions*, Phys. Rev. Lett. **98**, 190504 (2007).

[11] R. Raussendorf, J. Harrington and K. Goyal, *Topological fault-tolerance in cluster state quantum computation*, New Journal of Physics **9**, 199 (2007).

[12] D. S. Wang, A. G. Fowler and L. C. L. Hollenberg, *Surface code quantum computing with error rates over 1%*, Phys. Rev. A **83**, 020302(R) (2011), `quant-ph/1009.3686`.

[13] A. G. Fowler, A. C. Whiteside and L. C. L. Hollenberg, *Towards practical classical processing for the surface code*, Phys. Rev. Lett. **108**, 180501 (2012), ArXiv:1110.5133.

[14] K. M. Svore, D. P. DiVincenzo and B. M. Terhal, *Noise threshold for a fault-tolerant two-dimensional lattice architecture*, Quantum Inf. Comput. **7**, 297 (2007).

[15] F. M. Spedalieri and V. P. Roychowdhury, *Latency in local, two-dimensional, fault-tolerant quantum computing*, Quantum Information & Computation **9**(7), 666 (2009).

[16] A. G. Fowler, A. M. Stephens and P. Groszkowski, *High threshold universal quantum computation on the surface code*, Phys. Rev. A **80**, 052312 (2009).

[17] A. G. Fowler, D. S. Wang and L. C. L. Hollenberg, *Surface code quantum error correction incorporating accurate error propagation*, Quant. Info. Comput. **11**, 8 (2011), `quant-ph/1004.0255`.

[18] G. Duclos-Cianci and D. Poulin, *Fast decoders for topological quantum codes*, Physical Review Letters **104**(5), 050504 (2010), doi:10.1103/PhysRevLett.104.050504, `quant-ph/0911.0581`.

[19] G. Duclos-Cianci and D. Poulin, *A renormalization group decoding algorithm for topological quantum codes* (2010), `quant-ph/1006.1362`.

[20] H. Bombin, R. S. Andrist, M. Ohzeki, H. G. Katzgraber and M. A. Martin-Delgado, *Strong resilience of topological codes to depolarization* (2012), `quant-ph/1202.1852`.

[21] J. R. Wootton and D. Loss, *High threshold error correction for the surface code* (2012), `quant-ph/1202.4316`.

[22] A. G. Fowler, A. C. Whiteside, A. L. McInnes and A. Rabbani, *Topological code Autotune* (2012).

[23] D. S. Wang, A. G. Fowler, C. D. Hill and L. C. L. Hollenberg, *Graphical algorithms and threshold error rates for the 2d colour code*, Quant. Info. Comp. **10**, 780 (2010), ArXiv:0907.1708.

[24] H. Bombin, G. Duclos-Cianci and D. Poulin, *Universal topological phase of 2D stabilizer codes* (2011), `arXiv:quant-ph/1103.4606`.

[25] A. J. Landahl, J. T. Anderson and P. R. Rice, *Fault-tolerant quantum computing with color codes* (2011), `quant-ph/1108.5738`.

[26] P. Sarvepalli and R. Raussendorf, *Efficient Decoding of Topological Color Codes* (2011), `quant-ph/1111.0831`.

[27] P. W. Shor, *Polynomial time algorithms for discrete logarithms and factoring on a quantum computer*, In *Algorithmic Number Theory: First International Symposium ANTS-I*, p. 289 (1994).

[28] A. G. Fowler, M. Mariantoni, J. M. Martinis and A. N. Cleland, *Surface codes: Towards practical large-scale quantum computation*, Phys. Rev. A **86**, 032324 (2012), doi:10.1103/PhysRevA.86.032324.

[29] S. Bravyi and A. Kitaev, *Universal quantum computation based on a magic states distillation* (2004), `quant-ph/0403025`.

[30] B. W. Reichardt, *Improved magic states distillation for quantum universality* (2004), `arXiv:quant-ph/0411036v1`.

[31] E. Campbell and D. Browne, *On the structure of protocols for magic state distillation* (2009), arXiv:0908.0838.

[32] J. Koch, T. M. Yu, J. Gambetta, A. A. Houck, D. I. Schuster, J. Majer, A. Blais, M. H. Devoret, S. M. Girvin and R. J. Schoelkopf, *Charge insensitive qubit design derived from the cooper pair box*, Physical Review A **76**, 042319 (2007).

[33] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. C. White, J. Y. Mutus, A. G. Fowler, B. Campbell, Y. Chen, Z. Chen *et al.*, *Superconducting quantum circuits at the surface code threshold for fault tolerance*, Nature **508**, 500 (2014).

[34] J. M. Chow, J. M. Gambetta, E. Magesan, D. W. Abraham, A. W. Cross, B. R. Johnson, N. A. Masluk, C. A. Ryan, J. A. Smolin, S. J. Srinivasan and M. Steffen, *Implementing a strand of a scalable fault-tolerant quantum computing fabric*, Nature Communications **5**, 4014 (2014), doi:https://doi.org/10.1038/ncomms5015.

[35] C.-Y. Lu, W.-B. Gao, O. Gühne, X.-Q. Zhou, Z.-B. Chen and J.-W. Pan, *Demonstrating anyonic fractional statistics with a six-qubit quantum simulator*, Physical Review Letters **102**, 030502 (2009).

[36] J. K. Pachos, W. Wieczorek, C. Schmid, N. Kiesel, R. Pohlner and H. Weinfurter, *Revealing anyonic features in a toric code quantum simulation*, New Journal of Physics **11**, 083010 (2009).

[37] G. Feng, G. Long and R. Laflamme, *Experimental simulation of anyonic fractional statistics with an nmr quantum-information processor*, Physical Review Letters **88**, 022305 (2013).

[38] A. J. Park, E. McKay, D. Lu and R. Laflamme, *Simulation of anyonic statistics and its topological path independence using a seven-qubit quantum simulator*, New Journal of Physics **18**, 043043 (2016).

[39] Y. P. Zhong, D. Xu, P. Wang, C. Song, Q. J. Guo, W. X. Liu, K. Xu, B. X. Xia, C.-Y. Lu, S. Y. Han, J.-W. Pan and H. Wang, *Emulating anyonic fractional statistical behavior in a superconducting quantum circuit*, Physical Review Letters **117**, 110501 (2016).

[40] C. Song, D. Xu, P. F. Zhang, J. W. Wang, Q. J. Guo, W. X. Liu, K. Xu, H. Deng, K. Q. Huang, D. N. Zheng, S.-B. Zheng, H. Wang *et al.*, *7 sc qubit circuit modeling approx two plaquettes of surface code and showing initial anyone statistics*, Physical Review Letters **121**, 030502 (2018).

[41] C. K. Andersen, A. Remm, S. Lazar, S. Krinner, N. Lacroix, G. J. Norris, M. Gabureac, C. Eichler and A. Wallraff, *Repeated quantum error detection in a surface code*, Nature Physics **16**, 875 (2020).

[42] Z. C. G. Q. AI), *Exponential suppression of bit or phase errors with cyclic error correction*, Nature **595**, 383 (2021).

[43] K. S. G. Q. AI), *Realizing topologically ordered states on a quantum processor*, Science **374**, 1237 (2021).

[44] J. F. Marques, B. M. Varbanov, M. S. Moreira, H. Ali, N. Muthusubramanian, C. Zachariadis, F. Battistel, M. Beekman, N. Haider, W. Vlothuizen, A. Bruno, B. M. Terhal *et al.*, *Logical-qubit operations in an error-detecting surface code*, Nature Physics (2021), doi:https://doi.org/10.1038/s41567-021-01423-9.

[45] S. Krinner, N. Lacroix, A. Remm, A. D. Paolo, E. Genois, C. Leroux, C. Hellings, S. Lazar, F. Swiadek, J. Herrmann, G. J. Norris, C. K. Andersen *et al.*, *Realizing repeated quantum error correction in a distance-three surface code*, arXiv (2021), arXiv:2112.03708.

[46] Y. Zhao et al., *Realizing an error-correcting surface code with superconducting qubits*, arXiv (2021), arXiv2112.13505v1.

[47] J. Kelly, R. Barends, A. G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, J. Y. Mutus, B. Campbell, Y. Chen, Z. Chen, B. Chiaro *et al.*, *State preservation by repetitive error detection in a superconducting quantum circuit*, Nature **519**, 66 (2015).

[48] C. D. Wilen, S. Abdullah, N. A. Kurinsky, C. Stanford, L. Cardani, G. D'Imperio, C. Tomei, L. Faoro, L. B. Ioffe, C. H. Liu, A. Opremcak, B. G. Christensen *et al.*, *Correlated charge noise and relaxation errors in superconducting qubits*, Nature **594**, 369 (2021).

[49] J. M. Martinis, *Saving superconducting quantum processors from decay and correlated errors generated by gamma and cosmic rays.*, npj Quantum Inf. **7**, 90 (2021).

[50] M. McEwen et al. (Google AI team), *Resolving catastrophic error bursts from cosmic rays in large arrays of superconducting qubits*, Nature Physics (2021), doi:https://doi.org/10.1038/s41567-021-01432-8.

[51] A. Einstein, B. Podolsky and N. Rosen, *Can quantum mechanical description of physical reality be considered complete*, Phys. Rev. **47**, 777 (1935).

[52] D. Howard, *Einstein on locality and separability*, Stud. Hist. Phil. Sci. **16**, 171 (1985).

[53] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press (2000).

[54] G. Chaucer, *The Canterbury Tales* (1476).

[55] J. Edmonds, *Paths, trees and flowers*, Canad. J. Math. **17**, 449 (1965).

[56] J. Edmonds, *Maximum matching and a polyhedron with 0,1-vertices*, J. Res. Nat. Bur. Standards **69B**, 125 (1965).

[57] A. G. Fowler, *Analytic asymptotic performance of topological codes* (2012), quant-ph/1208.1334.

[58] J. J. Sakurai, *Modern Quantum Mechanics*, Addison Wesley (1994).

[59] D. Gottesman, *The Heisenberg representation of quantum computers* (2008), quant-ph/9807006.

[60] A. G. Fowler, *Low-overhead surface code logical H* (2012), quant-ph/1202.2369.

[61] I. Bloch, J. Dalibard and W. Zwerger, *Many-body physics with ultra-cold gases*, Rev. Mod. Phys. **80**, 885 (2008).

[62] I. Bloch, J. Dalibard and S. Nascimbene, *Quantum simulations with ultracold quantum gases*, Nature Physics **8**, 267 (2012).

[63] D. Leibfried, R. Blatt, C. Monroe and D. J. Wineland, *Quantum dynamics of single trapped ions*, Rev. Mod. Phys. **75**, 281 (2003).

[64] R. Blatt and D. Wineland, *Entangled states of trapped atomic ions*, Nature **453**, 1008 (2008).

[65] D. Kielpinski, C. R. Monroe and D. J. Wineland, *Architecture for a large-scale ion-trap quantum computer*, Nature **417**, 709 (2002).

[66] M. A. Eriksson, M. Friesen, S. N. Coppersmith, R. Joynt, L. J. Klein, K. Slinker, C. Tahan, P. M. Mooney, J. O. Chu and S. J. Koester, *Spin-based quantum dot quantum computing in silicon*, Quantum Information Processing **3**, 133 (2004).

[67] Y. Makhlin, G. Schön and A. Shnirman, *Quantum-state engineering with Josephson-junction devices*, Rev. Mod. Phys. **73**, 357 (2001).

[68] M. H. Devoret and J. M. Martinis, *Implementing qubits with superconducting integrated circuits*, Quantum Inf. Process. **3**, 163 (2004).

[69] J. Q. You and F. Nori, *Superconducting circuits and quantum information*, Physics Today **58**, 42 (2005).

[70] J. Clarke and F. K. Wilhelm, *Superconducting quantum bits*, Nature **453**, 1031 (2008).

[71] R. J. Schoelkopf and S. M. Girvin, *Wiring up quantum systems*, Nature **451**, 664 (2008).

[72] D. P. DiVincenzo, *Fault-tolerant architectures for superconducting qubits*, Phys. Scr. **T137**, 014020 (2009).