

Large-Scale Topological Quantum Computing with and without Majorana Fermions



Dissertation

zur Erlangung des Grades eines
Doktors der Naturwissenschaften

am Fachbereich Physik
der Freien Universität Berlin

vorgelegt von

Daniel Litinski

Berlin, August 2019

Erstgutachter: Prof. Felix von Oppen, PhD
Zweitgutachter: Prof. Dr. Jens Eisert

Tag der Disputation: 15.08.2019

Selbstständigkeitserklärung

Hiermit erkläre ich, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder Ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

18.03.2019

Daniel Litinski

Contents

Abstract	1
Zusammenfassung	3
List of Publications	5
Introduction	7
1 A Game of Surface Codes	23
2 Lattice Surgery with a Twist	61
3 Quantum Computing with Majorana Fermion Codes	79
4 Color Codes in Tetron-Based Architectures	107
5 Color Codes in Braiding-Based Networks	119
Outlook	147
Acknowledgments	149
Bibliography	149

Abstract

Quantum computers are devices that can solve certain problems faster than ordinary, classical computers. The fundamental units of quantum information are qubits, superpositions of two states $|0\rangle$ and $|1\rangle$. There are various approaches to construct such two-level systems, among others, using superconducting circuits, trapped ions or photons. A common feature of these physical systems is that their coherence times are relatively short compared to the length of useful computations. Superconducting qubits, for instance, are currently the most advanced solid-state qubits, but they decohere after $\sim 100 \mu\text{s}$, and any information stored in these qubits is lost. On the other hand, useful quantum computations may require quantum information to survive on time scales that are many orders of magnitude longer, as their runtimes can reach several hours or even days. Topological quantum computing is an approach to construct qubits that survive for the entire duration of such a long computation.

Topological quantum computing comes in two flavors. The condensed matter approach is to build error-resilient qubits using exotic quasiparticles in topological materials, most prominently Majorana zero modes in topological superconductors. Even though no such qubit has been built to date, the hope is that their coherence times may be significantly longer than the coherence times of currently available solid-state qubits, but are still expected to be too short for large-scale quantum computing. The quantum information approach is to combine many error-prone qubits to build more robust logical qubits using topological error-correcting codes, e.g., surface codes. Even though the first approach is hardware-based and the second approach is software-based, they are deeply related. With Majorana-based qubits, the main logical operations are Majorana fermion parity measurements. By replacing Majorana-based qubits with surface-code patches and parity measurements with lattice-surgery operations, schemes for quantum computation with Majorana-based qubits or with surface codes can be identical.

In this thesis, we explore how to construct a large-scale topological fault-tolerant quantum computer that can perform useful quantum computations. Here, topological refers to the nature of the quantum error-correcting code, while the underlying hardware may be based on non-topological qubits, but could also be composed of Majorana-based qubits. We provide a complete picture of such a large-scale device, breaking down large quantum computations into logical qubits and logical operations, describing how these logical operations are performed on the level of physical qubits and physical gates, and finally discussing how these physical qubits can be pieced together in a Majorana-based system using topological superconducting nanowires.

Zusammenfassung

Quantencomputer sind Maschinen, die bestimmte Rechnungen schneller durchführen können als herkömmliche, klassische Computer. Die grundlegende Einheit der Quanteninformation ist das Qubit, eine Superposition zweier Zustände $|0\rangle$ und $|1\rangle$. Es existieren zahlreiche Herangehensweisen, solche Zwei-Niveau-Systeme mithilfe physikalischer Komponenten zu konstruieren, wie zum Beispiel auf Basis supraleitender Schaltkreise, Ionenfallen oder Photonen. Eine Eigenschaft solcher realistischer Systeme ist, dass deren Kohärenzzeiten sehr kurz sind verglichen mit der Länge einer nützlichen Quantenrechnung. So dekohärieren beispielsweise supraleitende Qubits – die aktuell fortgeschrittensten Festkörperqubits – nach bereits $\sim 100 \mu\text{s}$, sodass jegliche in diesen Qubits gespeicherte Quanteninformation auf dieser Zeitskala verloren geht. Allerdings können nützliche Quantenrechnungen mehrere Stunden oder gar Tage dauern und benötigen daher Qubits, welche Quanteninformation deutlich länger erhalten können, als aktuell verfügbare Qubits. Eine Methode, solche extrem langlebigen Qubits zu konstruieren ist topologisches Quantenrechnen.

Hinter dem Begriff “topologisches Quantenrechnen” verbergen sich zwei unterschiedliche Herangehensweisen. Der Ansatz in der Festkörperphysik ist es, fehlerresistente Qubits mithilfe exotischer Quasiteilchen in topologischen Materialien zu konstruieren, wie beispielsweise anhand von Majorana-Nullmoden in topologischen Supraleitern. Obwohl bisherige Versuche, solche topologischen Qubits herzustellen, ohne Erfolg geblieben sind, erhofft man sich längere Kohärenzzeiten verglichen mit herkömmlichen Festkörperqubits, jedoch vermutlich nicht lang genug für nützliche Quantenrechnungen. Der Ansatz der Quanteninformationstheorie ist es, viele fehleranfällige Qubits zu resistenteren logischen Qubits mithilfe eines topologischen fehlerkorrigierenden Codes zu kombinieren, wie beispielsweise mittels Surface-Codes. Obwohl der erste Ansatz eher softwareorientiert und der zweite Ansatz eher hardwareorientiert ist, besteht ein enger Zusammenhang zwischen beiden Herangehensweisen. So kann man in beiden Fällen genau das gleiche Quantenrechsenschema verwenden, indem man Majorana-Qubits und deren primäre logischen Operationen – Messungen der Majorana-Fermionenparität – durch Surface-Code-Qubits und Lattice-Surgery-Operationen ersetzt.

In dieser Dissertation untersuchen wir, wie ein vollständiger topologischer fehlertoleranter Quantencomputer konstruiert werden kann, welcher in der Lage ist, lange, nützliche Quantenrechnungen durchzuführen. Hierbei bezieht sich “topologisch” auf den fehlerkorrigierenden Code, wohingegen die zugrundeliegende Hardware sowohl auf nicht-topologischen, als auch auf Majorana-Qubits basieren kann. Wir formulieren eine vollständige Beschreibung des Quantencomputers, indem wir große Quantenrechnungen auf logische Qubits und logische Operationen herunterbrechen, diese logischen Operationen auf physikalische Qubits und physikalische Operationen zurückführen, und schließlich beschreiben, wie physikalische Qubits aus topologischen supraleitenden Nanodrähten zu Majorana-Qubits zusammengebaut werden können.

List of Publications

This cumulative dissertation is based on the following first-author publications:

- Daniel Litinski: *A Game of Surface Codes: Large-Scale Quantum Computing with Lattice Surgery*, Quantum 3, 128 (2019),
doi:10.22331/q-2019-03-05-128
- Daniel Litinski and Felix von Oppen: *Lattice Surgery with a Twist: Simplifying Clifford Gates of Surface Codes*, Quantum 2, 62 (2018),
doi:10.22331/q-2018-05-04-62
- Daniel Litinski and Felix von Oppen: *Quantum Computing with Majorana Fermion Codes*, Phys. Rev. B 97, 205404 (2018),
doi:10.1103/PhysRevB.97.205404
- Daniel Litinski and Felix von Oppen: *Braiding by Majorana Tracking and Long-Range CNOT Gates with Color Codes*, Phys. Rev. B 96, 205413 (2017),
doi:10.1103/PhysRevB.96.205413
- Daniel Litinski, Markus S. Kesselring, Jens Eisert and Felix von Oppen: *Combining Topological Hardware and Topological Software: Color Code Quantum Computing with Topological Superconductor Networks*, Phys. Rev. X 7, 031048 (2017),
doi:10.1103/PhysRevX.7.031048

Introduction

A computer is a device that takes input data and carries out a sequence of logical operations to generate output data. Suitable sequences of operations can be used to solve various problems like adding numbers, searching databases, or learning to drive cars autonomously. The number of logical operations that need to be performed to generate an output is governed by the complexity of the problem being solved. Problems that require a number of operations that scales polynomially or even logarithmically with the size of the input (e.g., the length of the input bit string) are considered to be efficiently solvable. On the other hand, if the number of operations scales exponentially or super-exponentially with the size of the input, the problem is considered intractable.

Ordinary, classical computers store information using bits, binary digits 0 or 1. They process input data through logic gates, boolean functions that map bit strings onto other bit strings. Using this approach, many common problems can be solved efficiently, such as basic arithmetic or sorting lists. However, there are many problems that are expected to be intractable, including factoring numbers and various optimization problems. That is not to say that these problems cannot be solved by a classical computer, but the runtime of algorithms with exponential time complexity quickly surpasses hundreds of years on realistic machines even for modest input sizes.

Quantum computers [1], just like classical computers, input bit strings and generate bit strings as outputs. Contrary to classical computers, quantum computers store information using qubits, quantum bits that are superpositions of a $|0\rangle$ state and a $|1\rangle$ state. They process information by performing quantum gates, unitary operations that map the multi-qubit state stored in the quantum computer onto a different multi-qubit state. Using this approach, some classically intractable problems can be solved using only polynomially many quantum gates. Most notably, this includes factoring co-primes using Shor's algorithm and the simulation of large, correlated quantum systems. Quantum simulation, in particular, has various industry-related applications in quantum chemistry and material science, among others.

A quantum computation is typically expressed through a quantum circuit, see Fig. 1. Such a circuit is a sequence of instructions that is read from left to right. Each line of the circuit corresponds to a qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. After initializing all qubits in the states specified to the left of each wire, the quantum gates corresponding to the boxes and symbols drawn on top of the wires are executed one after the other. An output is obtained by measuring the qubits in the Pauli- Z basis, also called the computational basis, i.e., measuring whether the qubits are in the $|0\rangle$ state or the $|1\rangle$ state.

A quantum computer is called universal, if, by executing gates in this manner, it can approximately implement any multi-qubit unitary transformation with arbitrary precision. This requires only a finite set of basic instructions called a universal gate set. One such set of gates is the so-called standard set [2] consisting of the Hadamard gate H , phase gate S , the T gate, and the

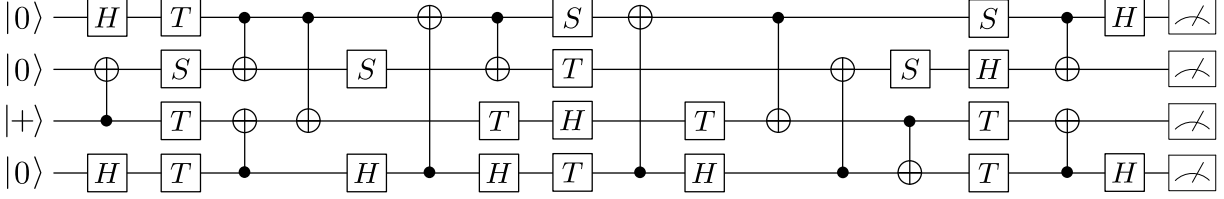


Figure 1: A random 4-qubit quantum circuit. Each line represents a qubit. The circuit is read from left to right with each box representing a quantum gate. Circles connected to crosses are CNOT gates, where the circle and cross indicate the control and target qubit, respectively. At the end of the circuit, the qubits are measured in the computational basis.

controlled-NOT gate CNOT. These gates correspond to the unitary matrices

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \quad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}, \quad \text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \quad (1)$$

The Hadamard gate interchanges $|0\rangle$ with $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$, and $|1\rangle$ with $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$. In this sense, this gate switches between the Pauli- Z and Pauli- X basis, where X , Y and Z are the Pauli operators

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \quad (2)$$

The S gate transforms $|+\rangle$ to $|\phi^+\rangle = (|0\rangle + i|1\rangle)/\sqrt{2}$ and $|-\rangle$ to $|\phi^-\rangle = (|0\rangle - i|1\rangle)/\sqrt{2}$. In this sense, the S gate switches between the X and Y basis. The CNOT gate is a two-qubit gate that can generate entanglement between qubits. The two qubits that it acts on are called control qubit and target qubit. The action of the CNOT gate is to flip the target qubit, if the control qubit is in the $|1\rangle$ state. Therefore, it maps $|00\rangle \rightarrow |00\rangle$, $|01\rangle \rightarrow |01\rangle$, $|10\rangle \rightarrow |11\rangle$, and $|11\rangle \rightarrow |10\rangle$, where the notation $|ij\rangle$ implies an outer product $|i\rangle \otimes |j\rangle$. The universal gate set is completed by the T gate, which is also the square root of the S gate. Gates that map Pauli eigenstates onto other Pauli eigenstates are called Clifford gates. Therefore, the standard set $\{H, T, S, \text{CNOT}\}$ is also referred to as Clifford+ T . Any quantum circuit can be pieced together using these four operations and measurements in the Z basis.

Useful, classically intractable quantum computations do not necessarily require many qubits. The smallest classically intractable quantum simulations use as few as 100 qubits [3, 4]. However, these useful computations tend to be very long, potentially consisting of billions of gates.

Overview: How to build a quantum computer

The main motivation of this thesis is to find efficient methods to translate such large-scale quantum computations into concrete hardware instructions, i.e., into physical gate operations. This is not a straightforward task, as physical qubits and physical gate operations are noisy and cannot be used directly to implement a large-scale quantum computation, since the noise would randomize the result of the computation and render it useless.

In this thesis, we discuss how to construct a large-scale quantum computer that can execute a 100-qubit billion-gate quantum computation. While there is not necessarily one definitive

approach to tackle this problem, the scheme discussed in this thesis is outlined in the following paragraphs. The basis of the quantum computer is an array of physical qubits – two-level systems – which can be assembled from different physical components. There exist various approaches [5] to encoding qubits in physical systems. Spin qubits use the two spin states of an electron confined to a quantum dot. Superconducting qubits use two states of a superconducting circuit, essentially the ground state and first excited state of a quantum anharmonic oscillator. Ion-trap qubits use two electronic states of cold trapped ions. Diamond qubits use the electron spin of a nitrogen-vacancy center in diamond. Optical qubits use the position or polarization of a photon. Majorana-based qubits use the fermion parity of Majorana fermion pairs. All these approaches differ with respect to coherence times, gate execution times and challenges concerning scalability. Moreover, all of these approaches are currently being pursued commercially, and it is not clear yet which platform is the most promising for large-scale quantum computing.

In a quantum computer, a sufficiently large number of physical qubits needs to be, e.g., arranged in a two-dimensional array, with some implementation of the previously mentioned single-qubit gates and two-qubit gates between nearest neighbors. 100 physical qubits are far from enough to run the 100-qubit computation, because these qubits are typically noisy. For instance, superconducting qubits have coherence times of $\sim 100 \mu s$, implying that any quantum information stored in these qubits is lost after hundreds of microseconds, whereas a billion-gate computation may take hours to finish.

Topological quantum computing is an approach to remedy this problem. In fact, topological quantum computing describes two approaches: a condensed-matter-based and a quantum-information-based approach. The condensed matter approach is to construct more error-resilient qubits using topological materials that are insensitive to local perturbation, most prominently using Majorana-based qubits constructed from topological superconductors [6, 7]. While no Majorana-based qubit has been successfully built to date, the hope is that their coherence times may be significantly higher compared to other solid-state qubits. Still, it is not expected that the coherence times of Majorana-based qubits will be long enough for hour-long quantum computations [8]. The quantum information approach is to combine many physical qubits – Majorana-based or otherwise – to more error-resilient logical qubits using a topological quantum error-correcting code [9, 10]. Quantum error correction enables the storage of qubits for, in principle, arbitrarily long times by periodically measuring certain check operators and using the measurement outcomes to detect and correct errors, provided that the error rate of the physical hardware is below a certain threshold. Topological codes, such as surface codes [9] or color codes [11], are families of codes that only require the measurement of geometrically local check operators, typically local in two dimensions, and feature significantly higher error thresholds compared to non-topological schemes.

To run a 100-qubit computation, it is therefore necessary to use hundreds of logical qubits, as opposed to physical qubits, which could amount to hundreds of thousands of physical qubits. Furthermore, the use of an error-correcting code restricts the set of available operations. Logical gates on logical qubits, in general, do not correspond to physical gates on physical qubits. With topological codes in two dimensions, which are well-suited for solid-state qubits, logical non-Clifford gates require the preparation of resource states, such as magic states [12] for the execution of logical T gates. For this reason, one possible way to construct a full fault-tolerant quantum computer is to partition it into blocks of logical qubits that are used to prepare these resource states (*distillation blocks*), and a block of logical qubits that stores the 100 qubits of the computation and consumes resource states to advance the computation (*data block*).

To reiterate, a large-scale quantum computer capable of performing classically intractable

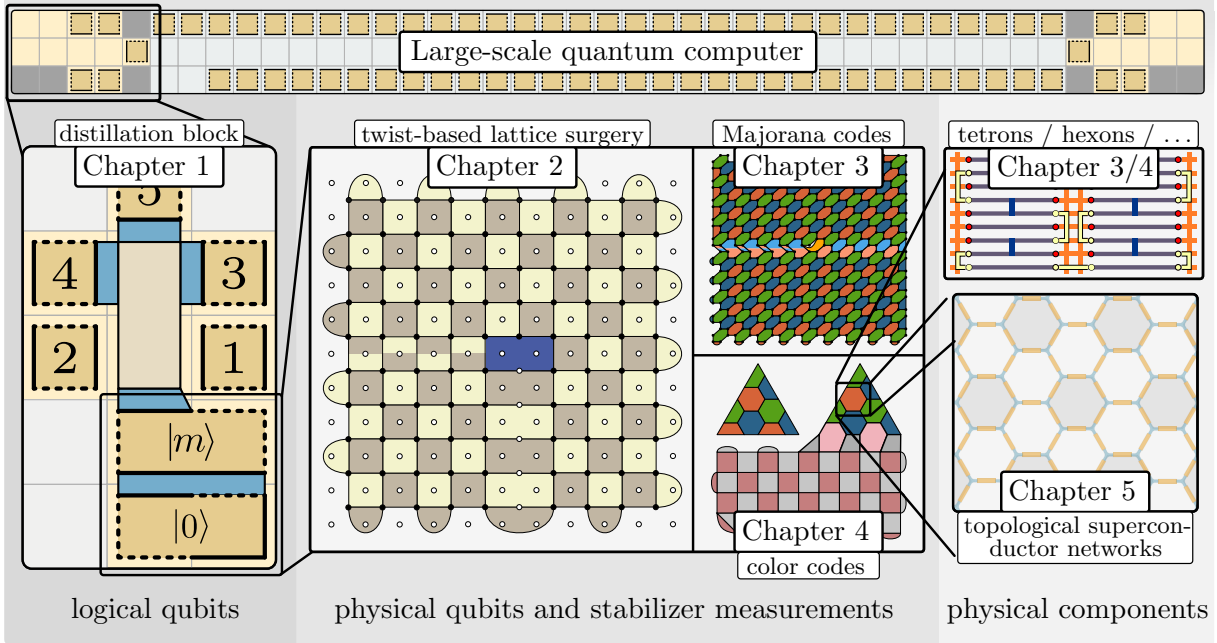


Figure 2: A large-scale quantum computer consists of data blocks and distillation blocks. The construction of these blocks in terms of logical qubits and logical operations is described in Chapter 1. The logical operations are logical Pauli product measurements. With surface codes, these correspond to twist-based lattice surgery, which is discussed in Chapter 2. Corresponding implementations with Majorana fermion codes and color codes are discussed in Chapters 3 and 4. In these chapters, we also show how to construct the required physical qubits and operations using tetron-like [13] Majorana-based qubits. Finally, in Chapter 5, we show how a fault-tolerant color-code-based quantum computer can be implemented in a braiding-based network [14] of topological superconductors.

computations can be constructed by piecing together physical ingredients to physical qubits, combining physical qubits to logical qubits using a topological error-correcting code, assembling many logical qubits to data and distillation blocks, and connecting many such blocks to a full quantum computer. As shown in Fig. 2, this also describes the structure of the thesis. While the chronological order in which these problems were treated in the publications of this thesis is one in which we zoom out from microscopic details to full error-corrected quantum computers, the order in which these works are presented here is reversed. However, before we can properly motivate the research performed within this thesis, we first need to introduce the central concepts used in this thesis: surface codes, lattice surgery, color codes and Majorana-based qubits.

Surface codes and lattice surgery

The basic working principle of a quantum error-correcting code is to use multiple physical qubits to encode a single logical qubit and to periodically measure certain operators to detect and correct errors. These operators are called check operators, stabilizer generators, or simply stabilizers. If not too many qubits are affected by errors between two rounds of stabilizer measurement, the errors can be corrected and the logical information stored in the logical qubit is preserved. The code distance d is the minimum number of errors that need to occur between two rounds of error

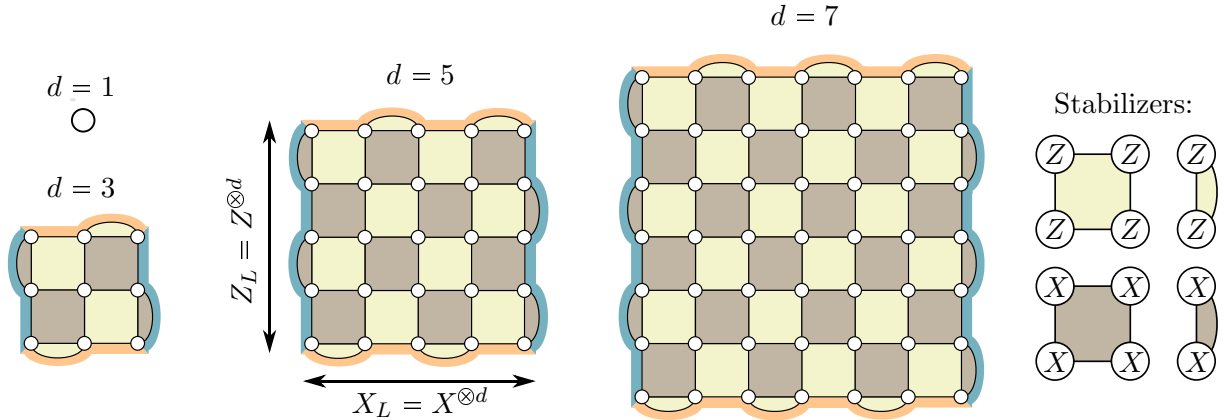


Figure 3: Physical qubit ($d = 1$) and examples surface-code patches with distances 3, 5 and 7.

corrections, such that the logical information is corrupted and the errors remain undetected. The higher the code distance, the more error-resilient the logical qubit, and the longer logical information can be stored before becoming corrupted.

Surface-code patches encode a single logical qubit using d^2 physical qubits, see Fig. 3. They are defined on a $d \times d$ square lattice, where each vertex corresponds to a physical qubit, and each face corresponds to a stabilizer operator that defines the code. Bright faces correspond to Z -type stabilizers $Z^{\otimes 4}$ or $Z^{\otimes 2}$ and dark faces are X -type stabilizers $X^{\otimes 4}$ or $X^{\otimes 2}$. For the example of $d = 5$, one logical qubit is encoded using 25 physical qubits. Since there are 24 mutually commuting stabilizers \mathcal{O}_i , the subspace of the Hilbert space with $\mathcal{O}_i = +1$ is spanned by two vectors, which are the logical $|0_L\rangle$ and $|1_L\rangle$ states. In other words, 25 physical qubits are 25 degrees of freedom and 24 stabilizer \mathcal{O}_i define 24 constraints $\mathcal{O}_i = +1$, leaving one remaining degree of freedom, which is the logical qubit. This can also be thought of as encoding the logical qubit in the doubly degenerate ground-state space of the Hamiltonian

$$H = - \sum_{i=1}^{24} \mathcal{O}_i. \quad (3)$$

It should be emphasized that this Hamiltonian is not to be interpreted as the Hamiltonian of a physical system, but rather as a measurement prescription. By projectively measuring the operators \mathcal{O}_i and enforcing a measurement outcome of $\mathcal{O}_i = +1$, the 25-qubit state is projected into the ground-state space of this Hamiltonian. Errors affecting the physical qubits will cause the system to leave this ground-state space. By periodically measuring all stabilizers, the system is projected back into the ground-state manifold. When errors occur, some stabilizers may be flipped, causing some measurement outcomes to be $\mathcal{O}_i = -1$ instead of $+1$. A sequence of Pauli operations can be used to flip all stabilizers back to $+1$. This sequence of operations is the error correction operation.

In the bulk of the code, each physical qubit is part of four stabilizers. At a boundary, qubits are only part of three stabilizers. We refer to boundaries where qubits are part of two X stabilizers and one Z stabilizer as Z boundaries. Conversely, at X boundaries, qubits are part of two X stabilizers and one Z stabilizer. Corners are points where two boundaries meet and qubits are part of only two stabilizers. Logical operators are operators that commute with all stabilizers, but are not stabilizers themselves. With surface codes, logical Z_L operators are strings of Z operators that go from one X boundary to another. Logical X_L operators are strings of X

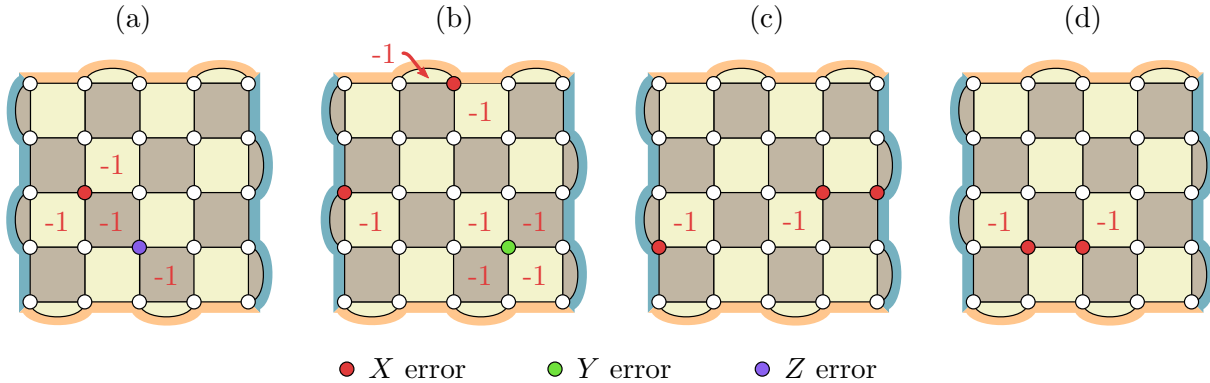


Figure 4: Examples of different error configurations and corresponding error syndromes on a distance-5 surface-code patch.

operators connecting two Z boundaries. In particular, a string of X 's (Z 's) along an X boundary (a Z boundary) is a logical X_L (Z_L) operator. For instance, a logical X_L operator of the $d = 7$ patch in Fig. 3 is the product $X^{\otimes 7}$ supported on the 7 qubits of any one of the X boundaries highlighted in orange. Storing the logical information in such a non-local string operator is what protects the encoded information from local errors.

Error detection and correction is performed in code cycles. In each code cycle, all stabilizer operators are measured. Consider an error model in which each physical qubit can be affected by a random Pauli error X , Y or Z in every code cycle with a probability p . While this is not necessarily a realistic noise model, the following considerations generalize to other local noise models. If a qubit in the bulk is affected by a Z error, as in Fig. 4a, the measurement outcome of the two X stabilizers that it is part of will flip from $+1$ to -1 in the subsequent code cycle, which can be used to detect and correct the error. Similarly, the measurement outcomes of Z stabilizers reveal X errors. Y errors (see Fig. 4b) correspond to simultaneous Z and X errors. At a boundary, errors may only flip one stabilizer. X errors will flip two Z stabilizers if located at an X boundary, but only one stabilizer at a Z boundary, as shown in Fig. 4b.

A specific configuration of errors will produce a certain pattern of stabilizer measurement outcomes called an error syndrome. However, only the syndrome is measured, while the underlying error configuration is unknown. In order to correct the errors, the error configuration needs to be determined from the syndrome. This classical computation is called decoding. For surface codes, various efficient decoding algorithms are known [15–20], even in the presence of stabilizer measurement errors, in which case stabilizer measurements need to be repeated to account for faulty measurement outcomes. It should be pointed out that, since the correction operations are always physical Pauli operations, they do not actually need to be performed in hardware, but can be classically tracked in software. Because the effect of Pauli corrections is to change the subsequent measurement outcomes of some stabilizers from $+1$ to -1 and vice versa – specifically, of those stabilizers that anticommute with the Pauli correction – one can simply reinterpret the outcomes of subsequent syndrome measurements, instead of actually performing the correction.

The problem that ultimately leads to logical errors is that multiple error configurations can produce identical error syndromes. For instance, the three X errors in Fig. 4c produce the same syndrome as the two X errors in Fig. 4d. Since only the syndrome is known, but not the error configuration, and two errors are more likely to occur than three, the correction operation in response to this syndrome would be to perform two X flips on the red qubits of Fig. 4d. This

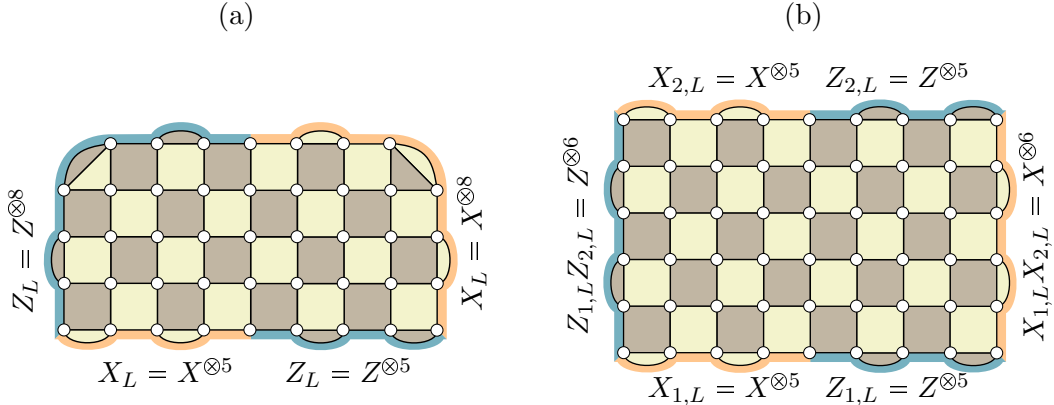


Figure 5: Examples of differently shaped surface-code patches.

successfully corrects the errors in the case of Fig. 4d, but leads to a logical X error in the case of Fig. 4c. In this sense, failure to successfully correct error strings that are longer than half the code distance leads to logical errors with surface codes. An error string that leads to logical errors in a surface code with odd distance d needs to affect at least $(d+1)/2$ qubits. Thus, in the regime of low p , surface codes suppress the error rate from p to $\sim p^{(d+1)/2}$.

Surface-code patches can have shapes that are different from the square patches shown in Fig. 3. One example with 43 physical qubits and 42 stabilizers is shown in Fig. 5a, where an X and Z boundary are on the same side of the patch. Since there is one more qubit than there are stabilizers, the patch encodes one logical qubit. Again, logical operators are string operators connecting two boundaries, such as Pauli strings along any of the highlighted boundaries. A patch with six boundaries is shown in Fig. 5b. Since it consists of 54 physical qubits and 52 stabilizers, it encodes two logical qubits instead of one. The logical operators corresponding to the X and Z Pauli strings located at the three X and Z boundaries of the patch can be defined as encoding the logical $X_{1,L}$ and $Z_{1,L}$ operators of the first logical qubit in the bottom two boundaries, and the logical $X_{2,L}$ and $Z_{2,L}$ operators of the second logical qubits in the top two boundaries. Satisfying the correct commutation relation, the left Z boundary corresponds to the product $Z_{1,L} \cdot Z_{2,L}$ and the right X boundary to $X_{1,L} \cdot X_{2,L}$. In general, a surface-code patch with n corners has $(n-1)/2$ more physical qubits than stabilizers, and therefore encodes $(n-1)/2$ logical qubits.

While stabilizer measurements and decoding are sufficient to fault-tolerantly store a qubit, quantum computation requires operations to manipulate the logical information encoded by surface codes. Surface-code qubits can be initialized in the logical $|0_L\rangle$ state by initializing all physical qubits in $|0\rangle$ and measuring all X stabilizers. The reason why this initializes the logical $|0_L\rangle$ state is that initializing all physical qubits in $|0\rangle$ sets all Z stabilizers to $+1$ and the logical Z_L operator to $+1$. The only requirement missing for a logical $|0_L\rangle$ state is for all X stabilizers to be $+1$, which is why these stabilizers are measured. Since the measurement outcomes are random, corrective Pauli flips are necessary to flip all X stabilizers that are -1 . These corrections are physical Z operations, which commute with the logical Z_L operator and therefore leave the encoded information unaffected. Similarly, a surface-code qubit can be initialized in the logical $|+_L\rangle$ state by initializing all physical qubits in $|+\rangle$ and measuring all Z stabilizers.

One useful type of logical operation will turn out to be the measurement of logical Pauli products $P_i \otimes P_j$ between neighboring logical qubits, where P_i (P_j) is a logical Pauli operator supported on logical qubit i (qubit j). Such measurements can be used to generate entanglement between

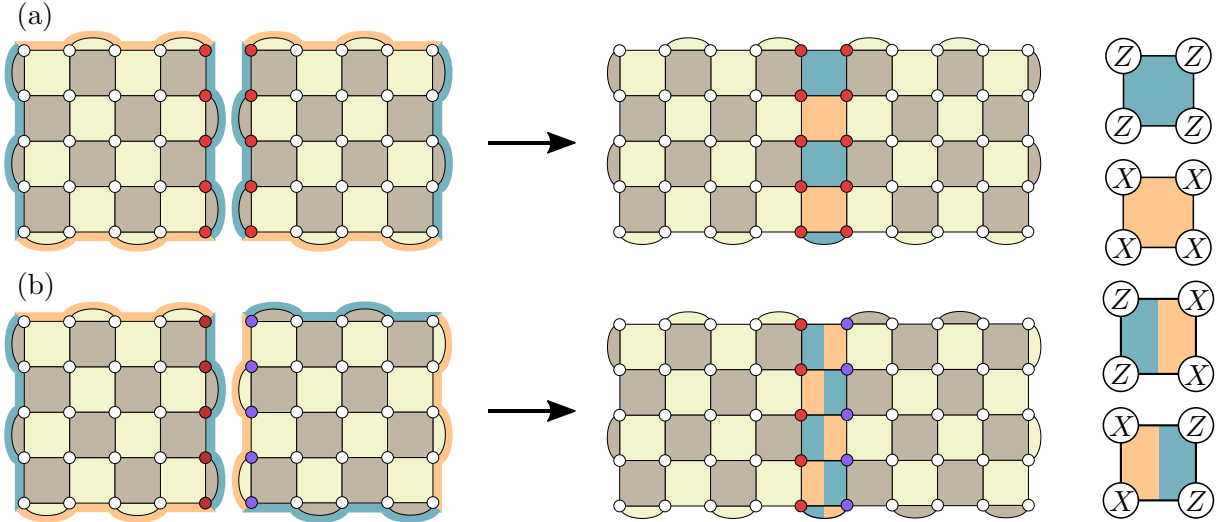


Figure 6: Example of a logical $Z_L \otimes Z_L$ (a) and $Z_L \otimes X_L$ (b) measurement via lattice surgery.

logical qubits, as, for instance, a $Z_L \otimes Z_L$ measurement between two logical qubits $|+_L\rangle \otimes |+_L\rangle$ with an outcome $+1$ will prepare an entangled logical Bell pair $(|0_L\rangle \otimes |0_L\rangle + |1_L\rangle \otimes |1_L\rangle)/\sqrt{2}$. Since logical operators are high-weight non-local operators, a specialized measurement protocol called *lattice surgery* [21] is employed to measure these products using only local, low-weight measurements. The simplest example is a $Z_L \otimes Z_L$ measurement, as shown in Fig. 6a. In terms of physical qubits, the operator $Z_L \otimes Z_L$ is the product of 10 Z operators $Z^{\otimes 10}$ supported on the 10 qubits highlighted in red. Note that, in Figs. 6 and 7, the highlighted qubits do not signify errors as in Fig. 4. To measure $Z_L \otimes Z_L$ using lattice surgery, the stabilizer configuration is changed by merging the neighboring boundary X stabilizers to form orange X stabilizers, and by introducing new Z stabilizers (blue stabilizers). In the new stabilizer configuration, the number of stabilizers has increased by one, implying that the number of degrees of freedom has decreased by one, and one bit of information has been measured. The measurement outcome of the orange stabilizers is trivial, since it is the product of previously measured stabilizers. The blue stabilizers yield non-trivial measurement outcomes. Their product corresponds to the $Z_L \otimes Z_L$ operator, which is the measured bit of information. The reason why this is a fault-tolerant measurement of this operator is that, in the second step, all stabilizers still commute and can therefore be used to detect and correct errors. In the presence of measurement errors, and assuming that measurement errors and qubit errors occur with similar error rates, the new stabilizer configuration is measured for d code cycles. Afterwards, the two patches are split again and the stabilizer configuration is reverted to the original configuration.

The interpretation of error syndromes during lattice surgery operations involves some subtleties. Because the blue stabilizers are unknown and determine the outcome of the $Z_L \otimes Z_L$ measurement, error correction after the merge is performed using a syndrome graph that does not involve the blue stabilizers, treating the punctures left behind in the lattice by disregarding these stabilizers as internal Z boundaries [22]. After the split, the two-qubit boundary stabilizers yield random, but correlated measurement outcomes. Their correction can lead to logical $Z_L \otimes Z_L$ “errors”, which do not affect the state, since it is a $Z_L \otimes Z_L$ eigenstate as a consequence of the $Z_L \otimes Z_L$ measurement.

A similar protocol can be used to measure the two-qubit operator $Z_L \otimes X_L$, as shown in Fig. 6b. This operator corresponds to the 10-qubit operator $Z^{\otimes 5} \otimes X^{\otimes 5}$, with the Z 's and X 's

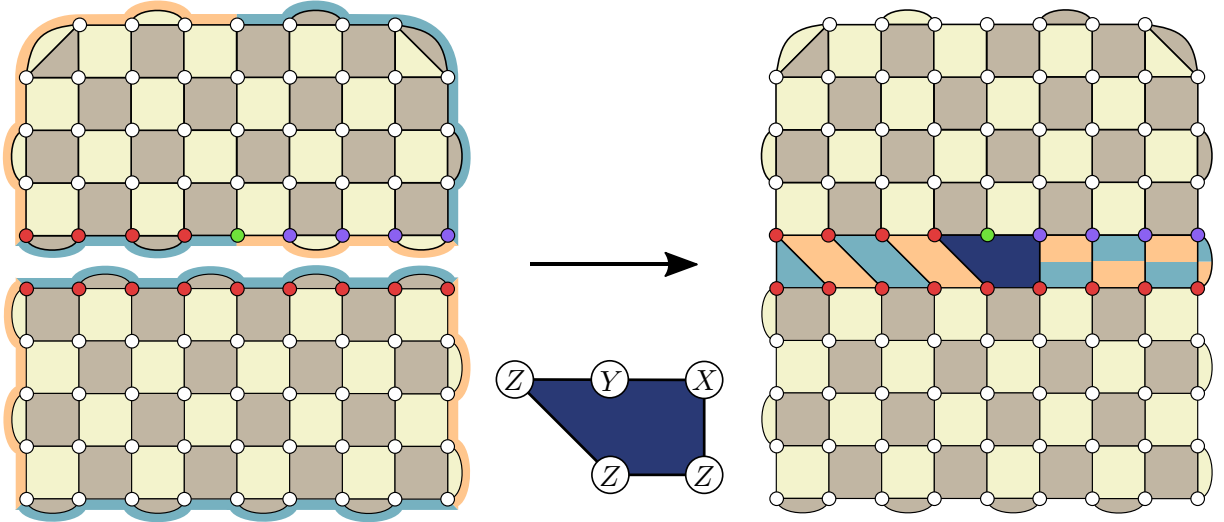


Figure 7: Example of a logical $Y_L \otimes Z_L$ measurement via twist-based lattice surgery.

supported on the qubits highlighted in red and purple, respectively. Again, boundary stabilizers are merged and new stabilizers are introduced. In this case, this yields mixed stabilizer of the form $X \otimes X \otimes Z \otimes Z$. The logical Pauli product is the product of the three non-trivial stabilizers.

Such a protocol can also be used to measure products involving the Y_L operator, as shown in Fig. 7. Since the Y_L operator of the top patch is the product $Y_L = iX_L Z_L$, it corresponds to the 9-qubit operator $Z^{\otimes 4} \otimes Y \otimes X^{\otimes 4}$, where the Z 's, Y and X 's are supported on the red, green and purple qubits, respectively. The bottom patch has a long 9-qubit Z boundary with a logical operator $Z_L = Z^{\otimes 9}$. To measure the logical Pauli product $Y_L \otimes Z_L$, lattice surgery is performed by merging boundary stabilizers to form two orange stabilizers and two mixed stabilizers, and introducing five non-trivial stabilizers whose product corresponds to $Y_L \otimes Z_L$. In this case, this involves a five-qubit operator (dark blue stabilizer). Since such an operator is also referred to as a *twist defect* [23], this lattice-surgery protocol is called *twist-based lattice surgery*, and it is introduced in Chapter 2.

As we discuss in Chapter 1, the ability to measure logical Pauli product operators is a computationally universal operation, if it is supplemented by the ability to prepare faulty resource states called magic states $|m\rangle = (|0\rangle + e^{i\pi/4}|1\rangle)/\sqrt{2}$. With surface codes, only logical Z and X Pauli eigenstates can be prepared fault-tolerantly. Arbitrary logical $|\psi_L\rangle$ can be prepared using an error-prone protocol called state injection. There are multiple variants of state injection [24, 25]. One simple approach similar to lattice surgery is to grow patches to full distance [21], as shown in Fig. 8. The protocol starts by initializing a distance-1 surface-code patch in the $|\psi_L\rangle$ state, i.e., a physical qubit in the $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ state. Next to it is a distance-1 four-qubit logical $|+_L\rangle$ state, which corresponds to a four-qubit GHZ state $(|0000\rangle + |1111\rangle)/\sqrt{2}$, essentially a logical $|+_L\rangle$ state using a repetition code. In the second step, the operator $Z \otimes Z$ between the qubit $|\psi\rangle$ and the GHZ state is measured. This can be interpreted as a $Z_L \otimes Z_L$ lattice surgery between two distance-1 patches. If the measurement outcome is +1, the state $|\psi\rangle \otimes |+_L\rangle$ is projected into the state $\alpha|00000\rangle + \beta|11111\rangle$, which is a logical $|\psi_L\rangle$ state in the repetition code, or, equivalently, a logical distance-1 surface-code patch in the $|\psi_L\rangle$ state. A measurement outcome of -1 differs by a Pauli correction. Below this patch, an additional 5×4 patch is initialized in the logical $|0_L\rangle$ state. Finally, in the third step, the patches are merged through an $X_L \otimes X_L$ measurement via lattice surgery. Similarly to the previous step,

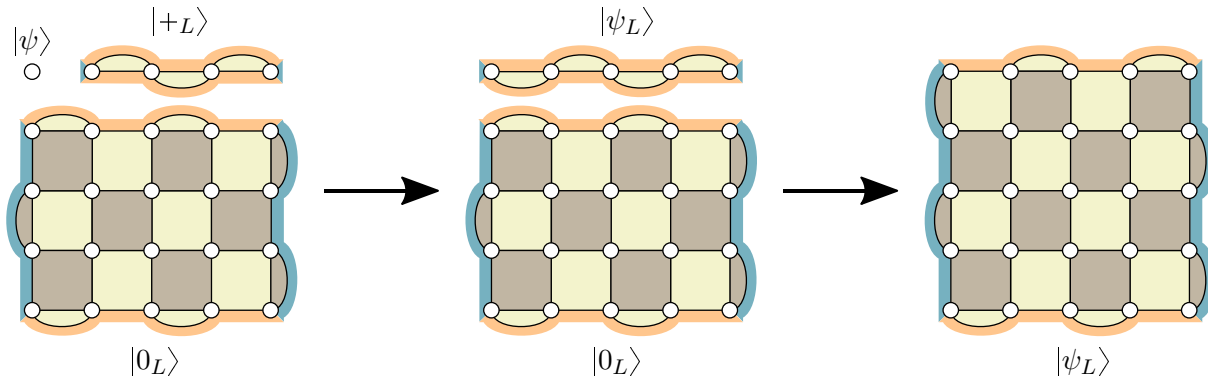


Figure 8: Example of a state-injection protocol to initialize a faulty $|\psi_L\rangle$ state.

this transfers the logical $|\psi_L\rangle$ state into the full distance-5 patch. Because the logical information is stored in a distance-1 patch in the first and second step, this protocol is not fault-tolerant. Therefore, it prepares a logical $|\psi_L\rangle$ state with an error rate proportional to the physical error rate p .

Color codes

Another family of topological quantum error-correcting codes that are closely related to surface codes are color codes [11]. These codes are defined on lattices whose faces can be colored in three different colors, such that no neighboring faces share the same color. One example is the hexagonal lattice shown in Fig. 9. Again, vertices correspond to physical qubits and the red, green and blue faces correspond to the stabilizers of the code. Regardless of the color, each stabilizer corresponds to an X -type and a Z -type stabilizer. Specifically, the codes shown in Fig. 9 feature $X^{\otimes 6}$ and $Z^{\otimes 6}$ stabilizers on the 6-qubit faces, and $X^{\otimes 4}$ and $Z^{\otimes 4}$ stabilizers on the 4-qubit faces.

Boundaries of color-code patches are introduced in a similar way as with surface codes. However, for color codes, we consider three different types of boundaries instead of two, namely red, green and blue boundaries. Qubits in the bulk are part of three different-colored faces (i.e., six stabilizers): red, green and blue. At a red boundary, physical qubits are only part of two faces, namely blue and green faces. Similarly, at blue (green) boundaries, qubits are part of red and green (red and blue) faces. Finally, at corners, qubits are part of only one face.

Logical qubits correspond to triangular color-code patches with three different-colored boundaries. The logical operators are strings of X 's and Z 's connecting all three boundaries. In particular, the product of all X operators $X^{\otimes d}$ along *any* of the three boundaries is a logical X operator. Similarly, the product of all Z operators $Z^{\otimes d}$ along *any* of the three boundaries is a logical Z operator.

The special property of color codes is that, contrary to surface codes, the support of X and Z stabilizers is identical, i.e., they overlap. Similarly, the support of X and Z logical operators is identical. Because $Y = iXZ$, this implies that Y stabilizers are also valid stabilizers of the code, and that the logical Y_L information corresponds to a similar string $Y^{\otimes d}$ of physical Y operators along any of the three boundaries. Furthermore, this implies that physical gates that map Paulis onto other Paulis – Clifford gates – must be transversal, meaning that applying a specific physical Clifford gate on all physical qubits is equivalent to the corresponding logical Clifford gate, which is not the case for surface codes. For instance, with color codes, logical Hadamard and S gates

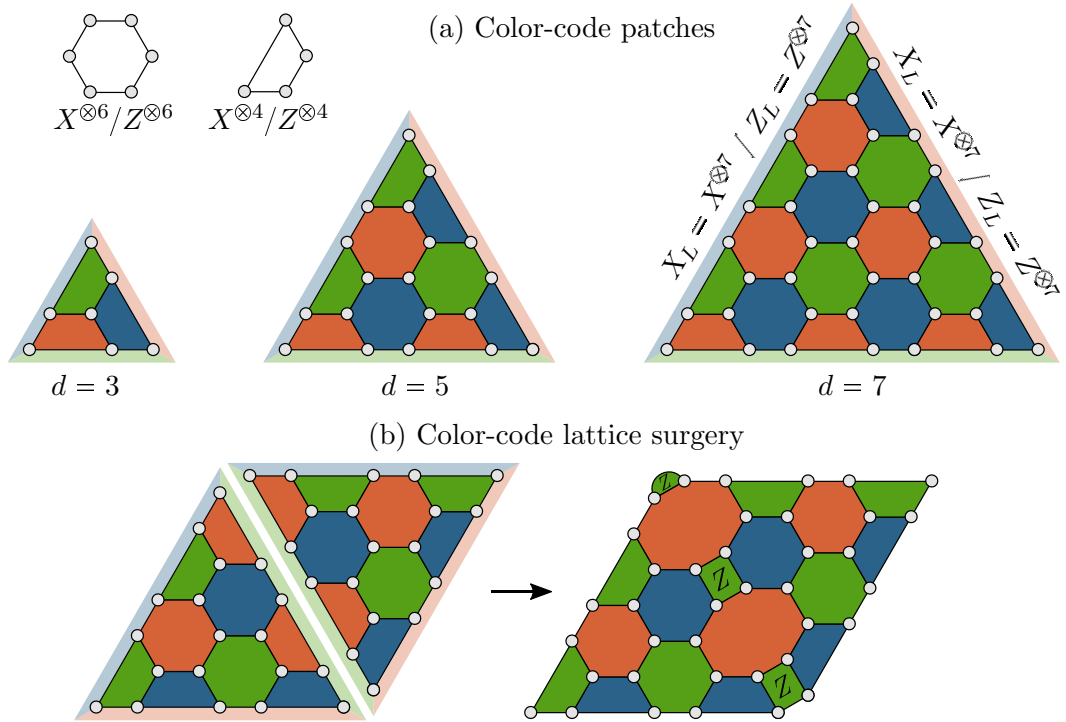


Figure 9: (a) Triangular color-code patches with distances $d = 3, 5$ and 7 . Each face corresponds to two stabilizers: an X -type and a Z -type stabilizer. (b) Example of a lattice-surgery protocol between two color-code patches for the measurement of the $Z_L \otimes Z_L$ operator.

can be implemented by performing physical Hadamard or S gates on all physical qubits of the color-code patch.

Lattice surgery can be performed using a similar protocol as for surface codes [24]. An example of a $Z_L \otimes Z_L$ measurement between two triangular color-code patches with neighboring green boundaries is shown in Fig. 9b. The red stabilizers are merged to yield the trivial stabilizers, and new green stabilizers are introduced as non-trivial check operators. Their product corresponds to the logical parity measurement, which is why these green stabilizers are only measured in the Z basis for the case of a $Z_L \otimes Z_L$ measurement.

To summarize, color codes are a family of topological codes in two dimensions which feature transversal Clifford gates. However, they also have higher-weight stabilizers compared to surface codes.

Majorana-based qubits

Another concept that will be relevant in Chapters 3-5 of this thesis are Majorana-based qubits, the generalization of which are Majorana fermion codes. Their building blocks are described by Majorana fermion operators γ_i , which are self-adjoint operators $\gamma_i = \gamma_i^\dagger$ that satisfy the fermionic anticommutation relations $\{\gamma_i, \gamma_j\} = 2\delta_{i,j}$. Majorana fermions are predicted to emerge as quasiparticles in various systems exhibiting a topological phase [7, 27–33], but none have been conclusively detected in experiments so far. For the purpose of Majorana fermion codes, the underlying origin of the Majorana fermions is unimportant, although their implementation in the quantum-wire-based architecture proposed in Ref. [13] will be outlined in Chapter 3.

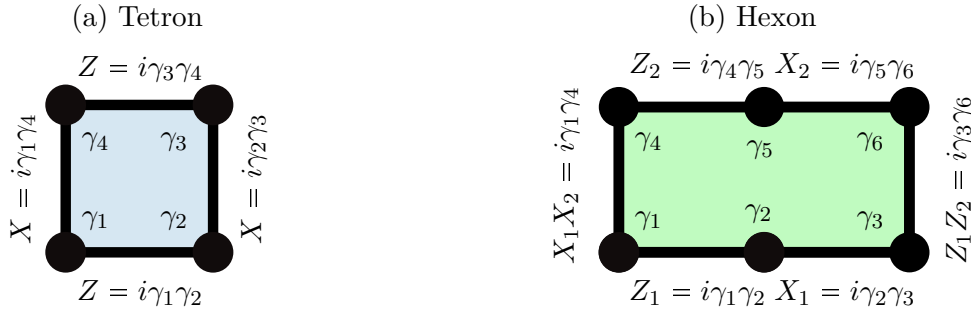


Figure 10: The smallest Majorana fermion codes are the four-Majorana code (tetron) and the six-Majorana code (hexon).

Even without error correction, a Majorana fermion code is necessary to encode a qubit with Majorana fermions. Contrary to physical qubits, single Majorana fermions have no measurable degrees of freedom, as fermion parity conservation dictates that all physically observable quantities must involve an even number of Majorana fermion operators. In contrast to complex fermions described by fermionic operators \hat{c} and \hat{c}^\dagger , Majorana fermions have no fermionic occupation number similar to $\hat{c}^\dagger\hat{c}$, since $\gamma^\dagger\gamma = 1$. Instead, the *parity* of Majorana fermions can be measured, corresponding to a product of an even number of Majorana fermion operators. For two Majorana fermions, the parity can be related to a fermionic occupation number, since any complex fermion can be written in terms of two Majorana fermions as $\hat{c} = (\gamma_1 + i\gamma_2)/2$ and $\hat{c}^\dagger = (\gamma_1 - i\gamma_2)/2$ with the two Majorana fermions as the real and imaginary part of a complex fermion. The fermionic occupation number of this complex fermion is proportional to the Majorana fermion parity, since $\hat{c}^\dagger\hat{c} = (i\gamma_1\gamma_2 + 1)/2$. While the occupation number can be 0 or 1, the Majorana fermion parity $i\gamma_1\gamma_2$ can be +1 or -1, which makes it similar to a Pauli operator.

Two Majorana fermions are insufficient to encode a qubit. One reason is that two Majorana fermions correspond to a complex fermionic state that can be empty or occupied. While this is a two-level system, fermion parity conservation prohibits any superposition of these two states. Another explanation is that, since only products of even numbers of Majorana fermion operators can be measured, the only observable with two Majorana fermions γ_1 and γ_2 is $i\gamma_1\gamma_2$. This defines one Pauli operator (say, Z), but a qubit requires three Pauli operators X , Y and Z , otherwise it is just a classical bit.

The smallest number of Majorana fermions that can be used to define a qubit is four. Since the total Majorana fermion parity is fixed, this essentially defines the smallest Majorana fermion code, which is referred to as a *tetron* [13], see Fig. 10a. Here, the qubit is encoded in the doubly degenerate ground-state space of the Hamiltonian $H = \gamma_1\gamma_2\gamma_3\gamma_4$, which fixes the total Majorana fermion parity $-\gamma_1\gamma_2\gamma_3\gamma_4 = +1$. In addition to $Z = i\gamma_1\gamma_2$, a second Pauli operator $X = i\gamma_2\gamma_3$ can be defined. These are valid Pauli operators, since they anticommute and each square to the identity. The Y operator follows as $Y = i\gamma_1\gamma_3$. In addition, due to the fixed total parity, the identities $Z = i\gamma_1\gamma_2 = i\gamma_3\gamma_4$ and $X = i\gamma_2\gamma_3 = i\gamma_1\gamma_4$ hold.

The second-smallest Majorana fermion code uses six Majorana fermions and is referred to as a *hexon* [13], see Fig. 10b. Here, two qubits are encoded in the fourfold degenerate ground-state space of the Hamiltonian $H = -i\gamma_1\gamma_2\gamma_3\gamma_4\gamma_5\gamma_6$. Again, Majorana fermion parity operators can be used to define Pauli operators of these two qubits, satisfying the correct commutation relations. One possibility is to use the bottom three Majorana fermions to encode the first qubit as $Z_1 = i\gamma_1\gamma_2$ and $X_1 = i\gamma_2\gamma_3$, the top three Majorana fermions to encode the second qubit as

$Z_2 = i\gamma_4\gamma_5$ and $X_2 = i\gamma_5\gamma_6$, which leaves the left and right Majorana fermions to correspond to $X_1 \cdot X_2 = i\gamma_1\gamma_4$ and $Z_1 \cdot Z_2 = i\gamma_3\gamma_6$.

The Majorana-based qubits defined as tetrons and hexons in Fig. 10 strongly resemble the definition of logical qubits in four-corner and six-corner surface-code patches in Figs. 3 and 5b, where Majorana fermions are replaced with corners of surface-code patches. Moreover, the logical operations between pairs of Majorana-based are measurements of fermion parity operators, e.g., measurements of $Z_A \otimes Z_B = (i\gamma_{A,1}\gamma_{A,2})(i\gamma_{B,1}\gamma_{B,2})$ between two tetrons A and B , which resemble lattice surgery. In fact, through this correspondence, computational schemes for surface-code patches and for Majorana-based qubits can be essentially identical. This correspondence is further explored in Chapter 3.

While it remains unclear whether Majorana-based qubits can offer longer coherence times compared to existing qubits, one defining property of Majorana-based qubits are their robust single-qubit Clifford gates. Conventional qubits can typically only be read out in the computational basis (say, the Z basis), whereas measurements in a different Pauli basis require the application of a single-qubit Clifford gate before the measurement. Since this gate can be noisy, the measurement outcome can be affected by the gate error. With Majorana-based qubits, Pauli measurements correspond to measurements of two-Majorana fermion parity operators $i\gamma_i\gamma_j$. Since no Pauli operator is distinctively different from the others, and since all Pauli operators can be read out using such a two-Majorana measurement, no single-qubit Clifford gates are required to change the measurement basis. Because this avoids a potentially noisy operation, the single-qubit Clifford gates of Majorana-based qubits are considered to be robust, meaning that Majorana-based qubits can be measured in all three Pauli bases.

Previous state of the art

The field of software-based topological quantum computing was initiated by Kitaev's seminal paper on the toric code [9] in 1997, which led to the development of surface codes by Bravyi and Kitaev [34] one year later. The field of hardware-based topological quantum computing was arguably also initiated by Kitaev in his seminal paper on unpaired Majorana fermions in quantum wires [6] in 2000, although it is predated by Freedman's work establishing the connection between quantum computing and topological matter [35]. Software-based and hardware-based topological quantum computing continued to be studied in a largely independent fashion.

Concrete quantum computing schemes with surface codes were developed by Dennis, Kitaev, Landahl and Preskill [36], although these early schemes envisioned a surface-code-based quantum computer as a stack of planar codes with logical operations performed via transversal gates. This is impractical for architectures where qubits are arranged in a two-dimensional array, as transversal two-qubit gates require non-local connections between qubits. Hole-defect-based encodings were first described by Raussendorf, Harrington and Goyal [37, 38] in 2005. These solved the problem of non-locality, but also implemented certain Clifford gates inefficiently, particularly the S gate, as these hole-based encodings offered no efficient way of manipulating the logical Y information. Color codes were developed by Bombin [11] in 2006. Bombin also established a connection between surface codes and Majorana fermions by describing twist defects [23] in 2010. One year later, in 2011, lattice surgery was developed by Horsman, Fowler, Devitt and Van Meter [21].

On the hardware side, the ongoing intensive experimental effort to realize Majorana-based qubits is largely motivated by theoretical work from 2010, when Oreg, Refael and von Oppen [39] and Lutchyn, Sau and Das Sarma [40] described how to realize topological superconductivity

in quantum wires based on superconductor-semiconductor heterostructures. Two years later, first experimental signatures of Majorana zero modes in such systems were reported by Mourik et al. [41] in 2012. Early proposals for quantum computing architectures with Majorana-based qubits focused on networks of Majorana wires where logical operations are performed by moving Majorana fermions for braiding. Notable proposals include architectures by Alicea et al. [42], van Heck et al. [43] and Aasen et al. [14]. With the proposal of Majorana box qubits and tetron/hexon qubits by Plugge et al. [44] and Karzig et al. [13] in 2016, the focus shifted towards architectures solely based on the measurement of Majorana fermion parity operators instead of the movement of Majorana fermions through a network of wires. In parallel, error correction schemes for Majorana qubits based on surface codes were developed by Vijay, Hsieh and Fu [45] and Landau et al. [46]. These schemes were based on hole-encoded surface-code qubits, and were therefore unable to benefit from the topologically protected single-qubit Clifford gates of Majorana-based qubits on the level of error-corrected logical qubits.

Motivation behind the publications comprising this thesis

My first contribution to the field was – together with my co-authors Markus Kesselring, Jens Eisert and Felix von Oppen – to develop an error correction scheme for Majorana-based qubits which manages to exploit the topologically protected single-qubit Clifford gates of Majorana-based qubits on the level of logical qubits. As previous schemes for error correction with Majorana qubits were based on hole-encoded surface codes, there existed no scheme that could benefit from the robust single-qubit Clifford gates of Majorana-based qubits on the logical level. In our work, this was achieved by combining a braiding-based network of Majorana nanowires in the spirit of Aasen et al. with topological color codes. While Majorana-based qubits feature robust single-qubit Clifford gates, color codes feature transversal single-qubit Clifford gates. As described in Chapter 5, the transversal Clifford gates of color codes enable the use of physical braiding operations for logical gates, exploiting the topological protection of Majorana fermions on the logical level.

Since the experimental efforts shifted away from braiding-based networks and towards measurement-based architectures in the year of the publication of this paper, I also described how this scheme could be incorporated in tetron-based architectures in the subsequent publication, as described in Chapter 4. Remarkably, logical single-qubit Clifford gates require no hardware operations in this scheme, but instead are performed by purely classical tracking operations. While the fact that this is theoretically possible had been known for decades, as this is the content of the Gottesman-Knill theorem [47], it had so far not been exploited in the context of error-corrected Majorana-based quantum computing.

Continuing with the philosophy of exploiting the Gottesman-Knill theorem, I developed a lattice-surgery scheme that enables the tracking of logical Clifford gates with surface-code patches, which is the twist-based lattice surgery scheme discussed in Chapter 2. Since tracking gates classically is less costly than performing them in hardware, this helps further reduce the overhead of error correction. Moreover, this work described how to measure products of arbitrary pairs of logical operators using lattice surgery, whereas only $Z_L \otimes Z_L$ and $X_L \otimes X_L$ measurements were described in the previous literature.

Next, I developed new variants of surface codes specifically tailored towards Majorana-based qubits, exploring the trade-offs between different lattices of Majorana fermions, as discussed in Chapter 3. In this work, I also further investigated the connection between Majorana-based qubits and surface codes, concluding that there is not only a mathematical correspondence between

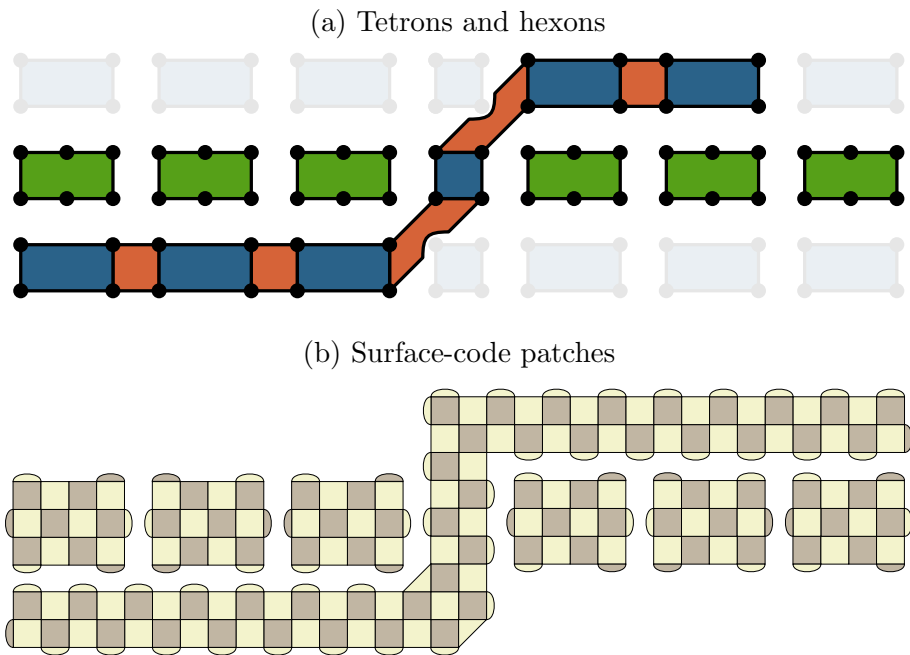


Figure 11: Comparison between an operation with Majorana-based tetrons and hexons (a) and surface-code patches (b). In this comparison, Majorana fermions correspond to corners of surface-code patches, and Majorana fermion parity measurements correspond to lattice-surgery operations.

twist defects and Majorana fermions, but also a pragmatic one, as computational schemes with Majorana-based qubits and surface-code patches can be essentially identical. Take for instance a Pauli product measurement protocol with tetrons and hexons described in Chapter 3, one step of which is shown in Fig. 11a. With surface codes, this protocol looks almost the same as the Majorana-based protocol, but with tetrons and hexons replaced with four-corner and six-corner surface-code patches, and Majorana fermion parity measurements replaced with lattice-surgery operations, as shown in Fig. 11b. In this sense, any computational protocol on Majorana-based qubits can be straightforwardly translated into a lattice-surgery protocol on surface-code patches.

Finally, I combined the previously developed logical lattice-surgery operations into a scheme to translate entire quantum computations into arrangements of surface-code patches and sequences of logical lattice-surgery measurements. This tackles the so-called *routing problem*, which deals with the arrangement of logical qubits in two dimensions and the scheduling of logical operations. While there exist prescriptions to convert quantum circuits into surface-code operations that predate my work [48–51], this conversion previously needed to be done with a computer and for each circuit individually, which made the conversion difficult to implement for large circuits of billions of gates acting on hundreds of qubits. The scheme presented in Chapter 1, on the other hand, is simple enough to be understood by humans without requiring complicated computations. By first translating circuits into a set of operations that are natural to surface codes, namely Pauli product rotations, hardware operations for logical Clifford gates are entirely avoided and instead tracked classically, reducing the overhead compared to previous schemes. In this sense, this work combines the protocols for logical operations developed in the previous publications into a scheme for large-scale quantum computation.

Structure of the thesis

The publications discussed in this thesis are not presented in a chronological order, but rather in an order where we zoom into a large-scale quantum computer. We start in Chapter 1, where we examine a large-scale quantum computer from a bird's-eye view, in terms of logical qubits and operations on logical qubits, disregarding the underlying physical qubits and error correction operations. The only concepts that are required to understand such a large-scale quantum computer are qubits and measurements, since the main operations of a quantum computer based on topological codes, such as surface codes, can be thought of as measurements of logical Pauli product operators. Next, in Chapter 2, we discuss how these logical Pauli product measurements can be performed with surface codes, where they correspond to a measurement protocol called lattice surgery. In Chapter 3, we move on to Majorana-based qubits and discuss how to design Majorana-specific variants of surface and color codes, and how to use these codes to perform lattice surgery. We find that, even though the condensed-matter and quantum-information approaches to topological quantum computing are conceptually different, schemes for quantum computing with Majorana-based qubits or with surface-code patches are almost identical, where lattice surgery can be used in the same way as Majorana fermion parity measurements. In Chapter 4, we discuss how the overhead of long-range communication between logical color-code qubits can be decreased by performing lattice surgery between surface codes and color codes. Finally, in Chapter 5, we zoom in one last time and piece together topological superconducting nanowires to a two-dimensional nanowire network through which Majorana zero modes can be moved. We use this to construct a braiding-based fault-tolerant quantum computer by adding fault-tolerance through topological color codes, which can use braiding operations to implement logical gates.

1 | A Game of Surface Codes

As we discussed in the introduction, Pauli product measurements and the preparation of faulty magic states can be used for fault-tolerant universal quantum computing. In the following publication, we discuss how large-scale quantum computations can be performed this way using lattice surgery.

A Game of Surface Codes: Large-Scale Quantum Computing with Lattice Surgery

Daniel Litinski @ Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, Arnimallee 14, 14195 Berlin, Germany

Given a quantum gate circuit, how does one execute it in a fault-tolerant architecture with as little overhead as possible? In this paper, we discuss strategies for surface-code quantum computing on small, intermediate and large scales. They are strategies for space-time trade-offs, going from slow computations using few qubits to fast computations using many qubits. Our schemes are based on surface-code patches, which not only feature a low space cost compared to other surface-code schemes, but are also conceptually simple – simple enough that they can be described as a tile-based game with a small set of rules. Therefore, no knowledge of quantum error correction is necessary to understand the schemes in this paper, but only the concepts of qubits and measurements.

The field of quantum computing is fuelled by the promise of fast solutions to classically intractable problems, such as simulating large quantum systems or factoring large numbers. Already ~ 100 qubits can be used to solve useful problems that are out of reach for classical computers [1, 2]. Despite the exponential speed-up, the actual time required to solve these problems is orders of magnitude above the coherence times of any physical qubit. In order to store and manipulate quantum information on large time scales, it is necessary to actively correct errors by combining many physical qubits into logical qubits using a quantum error-correcting code [3–5]. Of particular interest are codes that are compatible with the locality constraints of realistic devices such as superconducting qubits, which are limited to operations that are local in two dimensions. The most prominent such code is the surface code [6, 7].

Working with logical qubits introduces additional overhead to the computation. Not only is the space cost drastically increased as physical qubits are replaced by logical qubits, but also the time cost increases due to the restricted set of accessible logical operations. Surface codes, in particular, are limited to a set of 2D-local operations, which means that arbitrary gates in a quantum circuit may require several time steps instead of just one. To keep the cost of surface-code quantum computing low, it is important to find schemes that translate quantum circuits into surface-code layouts with a low space-time overhead. This is also necessary to benchmark how well quantum algorithms per-

form in a surface-code architecture.

There exist several encoding schemes for surface codes, among others, defect-based [7], twist-based [8] and patch-based [9] encodings. In this work, we focus on the latter. Surface-code patches have a low space overhead compared to other schemes, and offer low-overhead Clifford gates [10, 11]. In addition, they are conceptually less difficult to understand, as they do not directly involve braiding of topological defects. Designing computational schemes with surface-code patches only requires the concepts of qubits and measurements. To this end, we describe the operations of surface-code patches as a tile-based game. This is helpful to design protocols and determine their space-time cost. The exact correspondence between this game and surface-code patches is specified in Appendix A, but it is not crucial for understanding this paper. Readers who are interested in the detailed surface-code operations may read Appendix A in parallel to the following section.

Surface codes as a game. The game is played on a board partitioned into a number of tiles. An example of a 5×2 grid of tiles is shown in Fig. 1. The tiles can be used to host *patches*, which are representations of qubits. We denote the Pauli operators of each qubit as X , Y and Z . Patches have dashed and solid edges representing Pauli operators. We consider two types of patches: one-qubit and two-qubit patches. One-qubit patches represent one qubit and consist of two dashed and two solid edges. Each of the two dashed (solid) edges represent the qubit’s X (Z) operator. While the square patch in Fig. 1a only occupies one tile, a one-qubit patch can also be shaped to, e.g., occupy three tiles (b). A two-qubit patch (c) consists of six edges and represents two qubits. The first qubit’s Pauli operators X_1 and Z_1 are represented by the two top edges, while

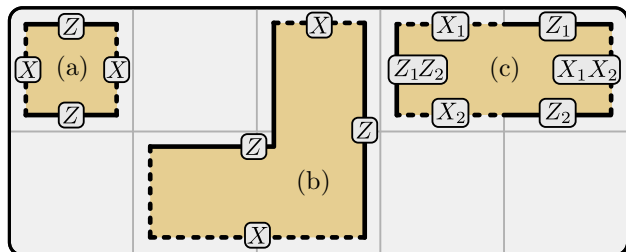


Figure 1: Examples of one-qubit (a/b) and two-qubit (c) patches in a 5×2 grid of tiles.

the second qubit's operators X_2 and Z_2 are found in the two bottom edges. The remaining two edges represent the operators $Z_1 \cdot Z_2$ and $X_1 \cdot X_2$.

In the following, we specify the operations that can be used to manipulate the qubits represented by patches. Some of these operations take one time step to complete (denoted by $1\ominus$), whereas others can be performed instantly, requiring $0\ominus$. The goal is to implement quantum algorithms using as few tiles and time steps as possible. There are three types of operations: qubit initialization, qubit measurement and patch deformation.

I. Qubit initialization:

- One-qubit patches can be initialized in the X and Z eigenstates $|+\rangle$ and $|0\rangle$. (Cost: $0\ominus$)
- Two-qubit patches can be initialized in the states $|+\rangle \otimes |+\rangle$ and $|0\rangle \otimes |0\rangle$. (Cost: $0\ominus$)
- One-qubit patches can be initialized in an arbitrary state. Unless this state is $|+\rangle$ or $|0\rangle$, an undetected random Pauli error may spoil the qubit with probability p . (Cost: $0\ominus$)

II. Qubit measurement:

- Single-patch measurements: The qubits represented by patches can be measured in the X or Z basis. For two-qubit patches, the two qubits must be measured simultaneously and in the same basis. This measurement removes the patch from the board, freeing up previously occupied tiles. (Cost: $0\ominus$)
- Two-patch measurements: If edges of two different patches are positioned in adjacent tiles, the product of the operators of the two edges can be measured. For example, the product $Z \otimes Z$ between two neighboring square patches can be measured, as highlighted in step 2 of Fig. 2a by the blue rectangle. If the edge of one patch is adjacent to multiple edges of the other patch, the product of all involved Pauli operators can be measured. For instance, if qubit A's Z edge is adjacent to both qubit B's X edge and Z edge, the operator $Z_A \otimes Y_B$ can be measured (see step 3 of Fig. 2d), since $Y = iXZ$. (Cost: $1\ominus$)
- Multi-patch measurements: An arbitrarily-shaped ancilla patch can be initialized. The product of any number of operators adjacent to the ancilla patch can be measured. The ancilla patch is discarded after the measurement. The example of a $Y_{|q_1\rangle} \otimes X_{|q_3\rangle} \otimes Z_{|q_4\rangle} \otimes X_{|q_5\rangle}$ measurement is shown in Fig. 2e. (Cost: $1\ominus$)

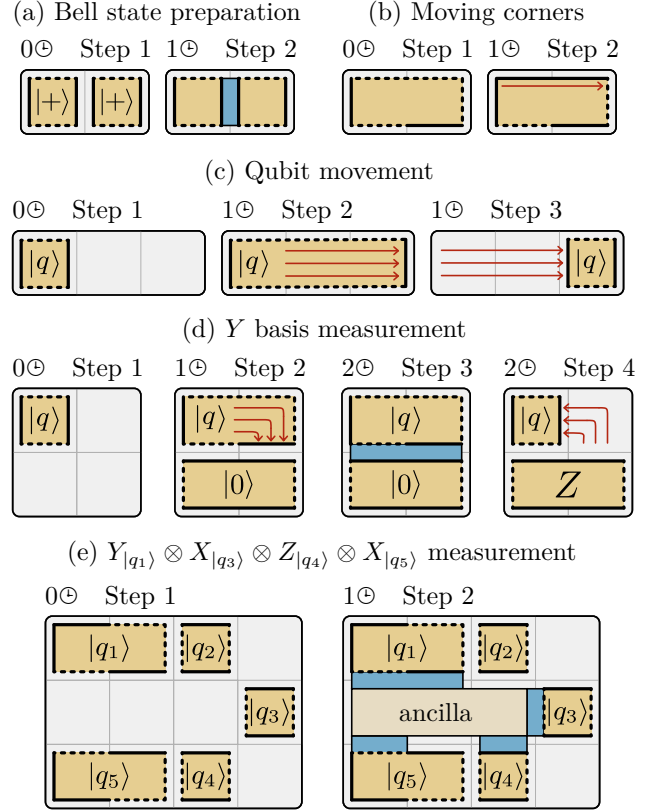


Figure 2: Examples of short protocols. (a) Preparation of a two-qubit Bell state in $1\ominus$. (b) Moving corners of a four-corner patch to change its shape in $1\ominus$. (c) Moving a square-patch qubit over long distances in $1\ominus$. (d) Measurement of a square-patch qubit in the Y basis using an ancilla qubit and $2\ominus$. (e) A multi-qubit $Y_{|q_1\rangle} \otimes X_{|q_3\rangle} \otimes Z_{|q_4\rangle} \otimes X_{|q_5\rangle}$ measurement in $1\ominus$.

III. Patch deformation:

- Edges of a patch can be moved to deform the patch. If the edge is moved onto a free tile to increase the size of the patch, this takes $1\ominus$ to complete. If the edge is moved inside the patch to make the patch smaller, the action can be performed instantly.
- Corners of a patch can be moved along the patch boundary to change its shape, as shown in Fig. 2b. (Cost: $1\ominus$)

To illustrate these operations, we go through three short example protocols in Fig. 2a/c/d. The first example (a) is the preparation of a Bell pair. Two square patches are initialized in the $|+\rangle$ state. Next, the operator $Z \otimes Z$ is measured. Before the measurement, the qubits are in the state $|+\rangle \otimes |+\rangle = (|00\rangle + |01\rangle + |10\rangle + |11\rangle)/2$. If the measurement outcome is $+1$, the qubits end up in the state $(|00\rangle + |11\rangle)/\sqrt{2}$. For the outcome -1 , the state is $(|01\rangle + |10\rangle)/\sqrt{2}$. In both cases, the two

qubits are in a maximally entangled Bell state. This protocol takes $1\ominus$ to complete. The second example (c) is the movement of a square patch into a different tile. For this, the square patch is enlarged by patch deformation, which takes $1\ominus$, and then made smaller again at no time cost. The third example (d) is the measurement of a square patch in the Y basis. For this, the patch is deformed such that the X and Z edge are on the same side of the patch. An ancillary patch is initialized in the $|0\rangle$ state and the operator $Z\otimes Y$ between the ancilla and the qubit is measured. The ancilla is discarded by measuring it in the Z basis.

Translation to surface codes. As described in Appendix A, protocols designed within this framework can be straightforwardly translated into surface-code operations. Essentially, patches correspond to surface-code patches with dashed and solid edges as rough and smooth boundaries. Thus, for surface codes with a code distance d , each tile corresponds to d^2 physical data qubits. Each time step roughly corresponds to d code cycles, i.e., measuring all surface-code check operators d times. We associate a time step with all surface-code operations which have a time cost that scales with d , but no time step with operations whose time cost is independent of the code distance, but may still be nonzero. For this reason, the correspondence between $1\ominus$ and d code cycles is not exact.

Two-patch and multi-patch measurements correspond to (twist-based) lattice surgery [9, 11] and multi-qubit lattice surgery [12], respectively, which both require d code cycles to account for measurement errors. Qubit initialization has no time cost, since, in the case of X and Z eigenstates, it can be done simultaneously with the subsequent lattice surgery [9, 13]. For arbitrary states, initialization corresponds to state injection [13, 14]. Its time cost does not scale with d . Similarly, single-qubit measurements in the X or Z basis correspond to the simultaneous measurement of all physical data qubits in the corresponding basis and some classical error correction, which does not scale with d either. Patch deformation is code deformation, which requires d code cycles, unless the patch becomes smaller in the process, in which case it corresponds to single-qubit measurements. Note that not all surface-code operations are covered by this framework. An extended set of rules is discussed in Appendix B.

In essence, the framework can be used to estimate the space-time cost of a computation. The leading-order term of the space-time cost – the term that scales with d^3 – of a protocol that uses s tiles for t time steps is $st \cdot d^3$ in terms of (physical data qubits)·(code cycles). The space cost is $s \cdot d^2$ physical data qubits. Determining the exact time cost requires special care. In some protocols, the subleading contributions due to state in-

jection and classical processing may need to be taken into account. For these protocols, we will show how they can be adapted to prevent such contributions from increasing the time cost beyond $t \cdot d$ code cycles.

Overview

Having established the rules of the game and the correspondence of our framework to surface-code operations, our goal is to implement arbitrary quantum computations. In this work, we discuss strategies to tackle the following problem: Given a quantum circuit, how does one execute it as fast as possible on a surface-code-based quantum computer of a certain size? This is an optimization problem that was shown to be NP-hard [15], so the focus is on heuristics rather than a general solution. The content of this paper is outlined in Fig. 3.

The input to our problem is an arbitrary gate circuit corresponding to the computation. We refer to the qubits that this circuit acts on as *data qubits*. As we review in Sec. 1, the natural universal gate set for surface codes is Clifford+ T , where Clifford gates are cheap and T gates are expensive. In fact, Clifford gates can be treated entirely classically, and T gates require the consumption of a magic state $|0\rangle + e^{i\pi/4}|1\rangle$. Only faulty (*undistilled*) magic states can be prepared in our framework. To generate higher-fidelity magic states for large-scale quantum computation, a lengthy protocol called magic state distillation [16] is used.

It is therefore natural to partition a quantum computer into a block of tiles that is used to distill magic states (a distillation block) and a block of tiles that hosts the data qubits (a data block) and consumes magic states. The speed of a quantum computer is governed by how fast magic states can be distilled, and how fast they can be consumed by the data block.

In Sec. 2, we discuss how to design data blocks. In particular, we show three designs: compact, intermediate and fast blocks. The compact block uses $1.5n + 3$ tiles to store n qubits, but takes up to $9\ominus$ to consume a magic state. Intermediate blocks use $2n + 4$ tiles and require up to $5\ominus$ per magic state. Finally, the fast block uses $2n + \sqrt{8n} + 1$ tiles, but requires only $1\ominus$ to consume a magic state. The compact block is an option for early quantum computers with few qubits, where the generation of a single magic state takes longer than $9\ominus$. The fast block has a better space-time overhead, which makes it more favorable on larger scales.

Data blocks need to be combined with distillation blocks for universal quantum computing. In Sec. 3, we discuss designs of distillation blocks. Since magic state distillation is the main operation of a surface-code-based quantum computer, it is important to minimize its space-time cost. We discuss distillation proto-

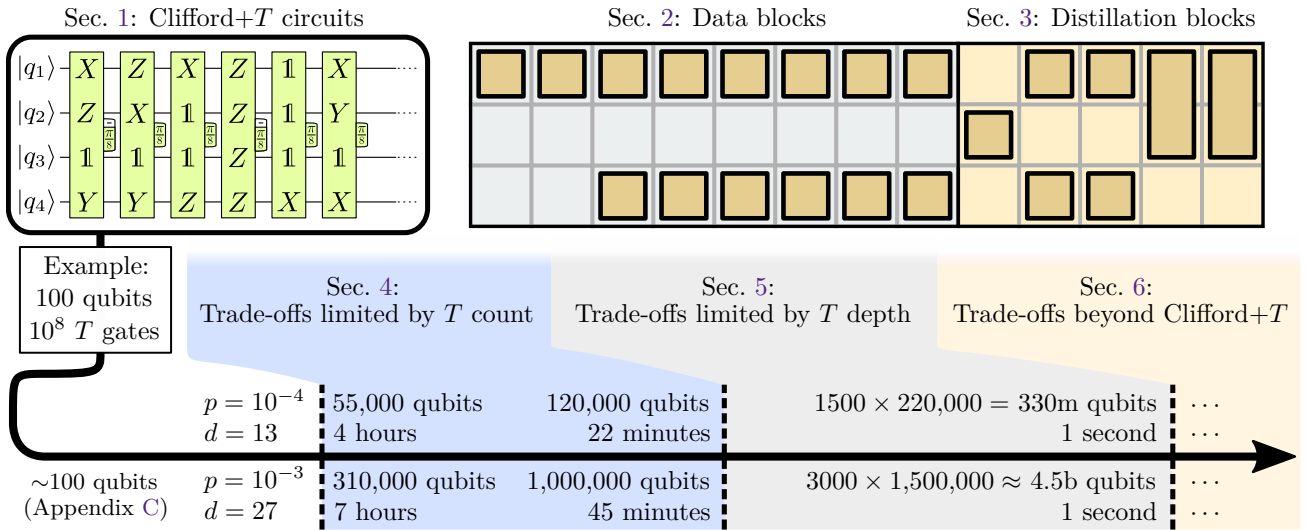


Figure 3: Overview of the content of this paper. To illustrate the space-time trade-offs discussed in this work, we show the number of physical qubits and the computational time required for a circuit of $10^8 T$ gates distributed over $10^6 T$ layers. We consider physical error rates of $p = 10^{-4}$ and $p = 10^{-3}$, for which we need code distances $d = 13$ and $d = 27$, respectively. We assume that each code cycle takes $1 \mu\text{s}$.

cols based on error-correcting codes with transversal T gates, such as punctured Reed-Muller codes [16, 17] and block codes [18–20]. In comparison to braiding-based implementations of distillation protocols, we reduce the space-time cost by up to 90%.

A data block combined with a distillation block constitutes a quantum computer in which T gates are performed one after the other. At this stage, the quantum computer can be sped up by increasing the number of distillation blocks, effectively decreasing the time it takes to distill a single magic state, as we discuss in Sec. 4. In order to illustrate the resulting space-time trade-off, we consider the example of a 100-qubit computation with $10^8 T$ gates, which can already be used to solve classically intractable problems [2]. Assuming an error rate of $p = 10^{-4}$ and a code-cycle time of $1 \mu\text{s}$, a compact data block together with a distillation block can finish the computation in 4 hours using 55,000 physical qubits.¹ Adding 10 more distillation blocks increases the qubit count to 120,000 and decreases the computational time to 22 minutes, using $1 \otimes$ per T gate.

For further space-time trade-offs in Sec. 5, we exploit that the T gates of a circuit are arranged in layers of gates that can be executed simultaneously. This enables linear space-time trade-offs down to the execution

¹We will assume that the total number of physical qubits is twice the number of physical data qubits. This is consistent with superconducting qubit platforms, where the use of measurement ancillas doubles the qubit count. If a platform does not require the use of ancilla qubits, the total qubit count is reduced by 50% compared to the numbers reported in this paper.

of one T layer per qubit measurement time, effectively implementing Fowler’s time-optimal scheme [21]. If the $10^8 T$ gates are distributed over 10^6 layers, and measurements (and classical processing) can be performed in $1 \mu\text{s}$, up to 1500 units of 220,000 qubits can be run in parallel, where each unit is responsible for the execution of one T layer. This way, the computational time can be brought down to 1 second using 330 million qubits. While this is a large number, the units do not necessarily need to be part of the same quantum computer, but can be distributed over up to 1500 quantum computers with 220,000 qubits each, and with the ability to share Bell pairs between neighboring computers.

In Sec. 6, we discuss further space-time trade-offs that are beyond the parallelization of Clifford+ T circuits. In particular, we discuss the use of Clifford+ φ circuits, i.e., circuits containing arbitrary-angle rotations beyond T gates. These require the use of additional resources, but can speed up the computation. We also discuss the possibility of hardware-based trade-offs by using higher code distances, but in turn shorter measurements with a decreased measurement fidelity. Ultimately, the speed of a quantum computer is limited by classical processing, which can only be improved upon by faster classical computing.

Finally, we note that while the number of qubits required for useful quantum computing is orders of magnitude above what is currently available, a proof-of-principle two-qubit device demonstrating all necessary operations using undistilled magic states can be built with 48 physical data qubits, see Appendix C.

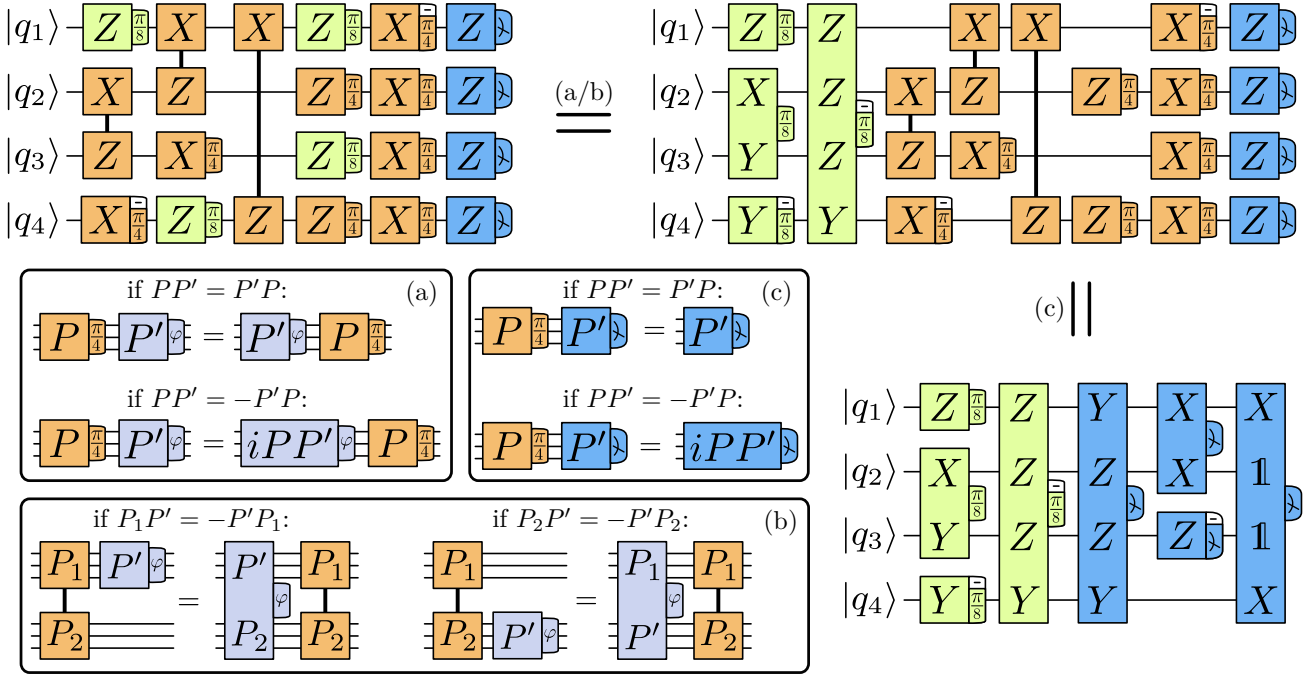


Figure 4: A generic circuit consists of $\pi/4$ rotations (orange), $\pi/8$ rotations (green) and measurements (blue). The Pauli product in each box specifies the axis of rotation or the basis of measurement. If the Pauli operator is $-P$ instead of P , a minus sign is found in the corner of the box, such that, e.g., $Z_{-\pi/4}$ corresponds to an S^\dagger gate. Using the commutation rules in (a/b), all Clifford gates can be moved to the end of the circuit. Using (c), the Clifford gates can be absorbed by the final measurements.

1 Clifford+ T quantum circuits

Our goal is to implement full quantum algorithms with surface codes. The input to our problem is the algorithm's quantum circuit. The universal gate set Clifford+ T is well-suited for surface codes, since it separates easy operations from difficult ones. Often, this set is generated using the Hadamard gate H , phase gate S , controlled-NOT (CNOT) gate, and the T gate. Instead, we choose to write our circuits using Pauli product rotations P_φ (see Fig. 5), because it simplifies circuit manipulations. Here, $P_\varphi = \exp(-iP\varphi)$, where P is a Pauli product operator (such as Z , $Y \otimes X$, or $X \otimes \mathbb{1} \otimes X$) and φ is an angle. In this sense, $S = Z_{\pi/4}$, $T = Z_{\pi/8}$, and $H = Z_{\pi/4} \cdot X_{\pi/4} \cdot Z_{\pi/4}$. The CNOT gate can also be written in terms of Pauli product rotations as $\text{CNOT} = (Z \otimes X)_{\pi/4} \cdot (\mathbb{1} \otimes X)_{-\pi/4} \cdot (Z \otimes \mathbb{1})_{-\pi/4}$. In fact, we can more generally define P_1 -controlled- P_2 gates as $C(P_1, P_2) = (P_1 \otimes P_2)_{\pi/4} \cdot (\mathbb{1} \otimes P_2)_{-\pi/4} \cdot (P_1 \otimes \mathbb{1})_{-\pi/4}$. The CNOT gate is the specific case of $C(Z, X)$.

Getting rid of Clifford gates. Clifford gates are considered to be easy, because, by definition, they map Pauli operators onto other Pauli operators [22]. This can be used to simplify the input circuit. A generic circuit is shown in Fig. 4, consisting of Clifford gates, $Z_{\pi/8}$ rotations and Z measurements. If all Clifford gates are

commuted to the end of the circuit, the $Z_{\pi/8}$ rotations become Pauli product rotations. The rules for moving $P_{\pi/4}$ rotations past P'_φ gates are shown in Fig. 4a: If P and P' commute, $P_{\pi/4}$ can simply be moved past P'_φ . If they anticommute, P'_φ turns into $(iPP')_\varphi$ when $P_{\pi/4}$ is moved to the right. Since $C(P_1, P_2)$ gates consist of $\pi/4$ rotations, similar rules can be derived as shown

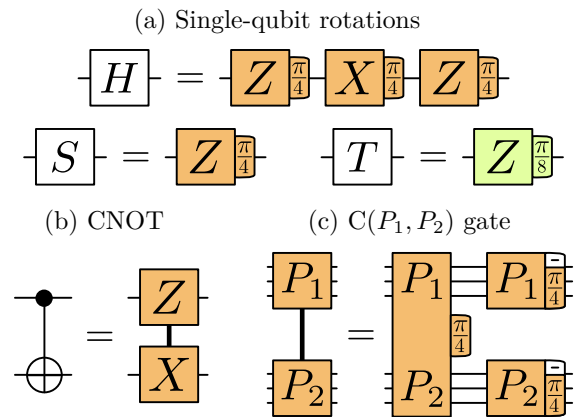


Figure 5: Clifford+ T gates in terms of Pauli rotations. (a) Single-qubit Clifford gates are $\pi/4$ rotations, and the T gate is a $\pi/8$ rotation. (b/c) P_1 -controlled- P_2 gates are Clifford gates, where $C(Z, X)$ is the CNOT gate.

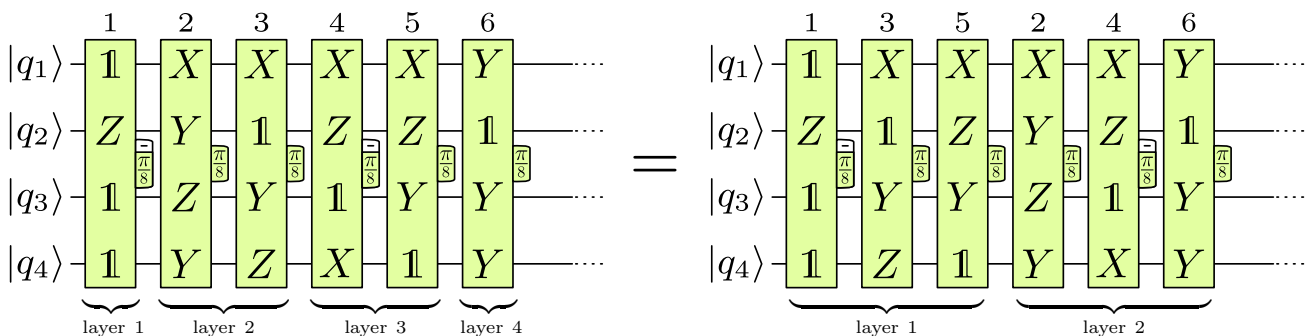


Figure 6: Clifford+ T circuits can be written as a number of consecutive $\pi/8$ rotations. These gates are grouped into layers of mutually commuting rotations. A simple greedy algorithm can be used to reduce the number of layers, i.e., the T depth.

in Fig. 4b: If P' anticommutes with P_1 , P'_φ turns into $(P'P_2)_\varphi$ after commutation. If P' anticommutes with P_2 , P'_φ turns into $(P'P_1)_\varphi$. If P' anticommutes with both P_1 and P_2 , P'_φ turns into $(P'P_1P_2)_\varphi$.

After moving the Clifford gates to the right, the resulting circuit consists of three parts: a set of $\pi/8$ rotations, a set of $\pi/4$ rotations, and Z measurements. Because Clifford gates map Pauli operators onto other Pauli operators, the Clifford gates can be absorbed by the final measurements, turning Z measurements into Pauli product measurements. The commutation rules of this final step are shown in Fig. 4c and are similar to the commutation of Clifford gates past rotations.

T count and T depth. Thus, every n -qubit circuit can be written as a number of consecutive $\pi/8$ rotations and n final Pauli product measurements, as shown in Fig. 6. We refer to the number of $\pi/8$ rotations as the T count. An important part of circuit optimization is the minimization of the T count, for which there exist various approaches [23–26]. The $\pi/8$ rotations of a circuit can be grouped into layers. All $\pi/8$ rotations that are part of a layer need to mutually commute. The number of $\pi/8$ layers of a circuit is strictly speaking not the same quantity as the T depth, but we will still refer to it as the T depth and to $\pi/8$ layers as T layers. Note

```

repeat
  for each layer  $i$  do
    for each rotation  $j$  in layer  $i + 1$  do
      if (rotation  $j$  commutes with all
        rotations in layer  $i$ ) then
        Move rotation  $j$  from layer  $i + 1$  to
          layer  $i$ ;
      end
    end
  end
until the partitioning no longer changes;
Algorithm to reduce the  $T$  count and  $T$  depth.

```

that, in the usual definition, only up to n T gates can be part of a layer, whereas in our case, there is no limit.

When partitioning $\pi/8$ rotations into layers, the naive approach often yields more layers than are necessary. For instance, a naive partitioning of the first 6 T gates of Fig. 6 yields 4 layers. A few commutations can bring the number down to 2 layers. There are a number of algorithms for the optimization of the T depth [27–29]. Here, we use the simple greedy algorithm shown below to reduce the number of layers.

Note that when a reordering puts two equal $\pi/8$ rotations into the same layer, they can be combined into a $\pi/4$ rotation that is commuted to the end of the circuit, thereby decreasing the T count. As we discuss in Sec. 6, this kind of algorithm can not only be used with $\pi/8$ rotations, but, in principle, with arbitrary Pauli product rotations. The reduction of the circuit depth in terms of non- $\pi/8$ rotations can be useful when going beyond Clifford+ T circuits.

1.1 Pauli product measurements

When implementing circuits like Fig. 6 with surface codes, one obstacle is that $\pi/8$ rotations are not directly part of the set of available operations. Instead, one uses magic states [16] as a resource. These states are $\pi/8$ -rotated Pauli eigenstates $|m\rangle = |0\rangle + e^{i\pi/4}|1\rangle$. They can be consumed in order to perform $P_{\pi/8}$ rotations. The corresponding circuit [30] is shown in Fig. 7.

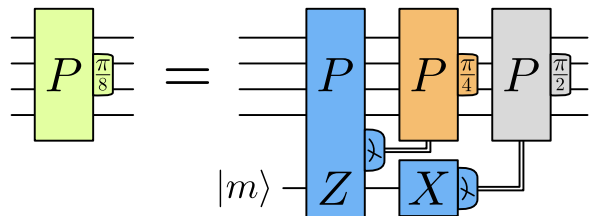


Figure 7: Circuit to perform a $\pi/8$ rotation by consuming a magic state.

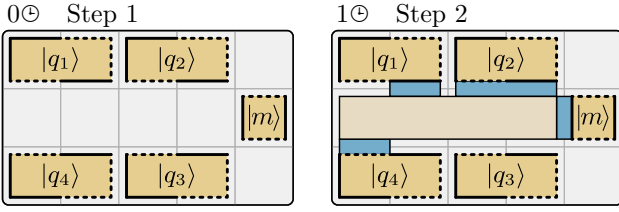


Figure 8: Example of a $Z_{|q_1\rangle} \otimes Y_{|q_2\rangle} \otimes X_{|q_4\rangle} \otimes Z_{|m\rangle}$ measurement to implement a $(Z \otimes Y \otimes \mathbb{1} \otimes X)_{\pi/8}$ gate.

A $P_{\pi/8}$ rotation corresponds to a $P \otimes Z$ measurement involving the magic state. If the measurement outcome is $P \otimes Z = -1$, then a corrective $P_{\pi/4}$ operation is necessary. Since this is a Clifford gate, it can be simply commuted to the end of the circuit, changing the axes of the subsequent $\pi/8$ rotations. Finally, in order to discard the magic state, it is disentangled from the rest of the system by an X measurement. Here, an outcome $X = -1$ prompts a $P_{\pi/2}$ correction. $\pi/2$ rotations correspond to Pauli operators, i.e., $P_{\pi/2} = P$. The Pauli correction can also be commuted to the end of the circuit. When $P_{\pi/2}$ is moved past a P' rotation or measurement, it changes the axis of rotation or measurement basis to $-P'$, if P and P' anticommute.

In essence, if magic states are available, the only operations required for universal quantum computing are Pauli product measurements. In our framework, such operations can be performed in $1 \odot$ via multi-patch measurements, corresponding to multi-qubit lattice surgery. An example is shown in Fig. 8, where a $(Z \otimes Y \otimes \mathbb{1} \otimes X)_{\pi/8}$ rotation on four qubits $|q_1\rangle$ - $|q_4\rangle$ stored in four two-tile one-qubit patches is performed. Using the circuit identity in Fig. 7, this is done by measuring $Z_{|q_1\rangle} \otimes Y_{|q_2\rangle} \otimes X_{|q_4\rangle} \otimes Z_{|m\rangle}$ between the four qubits and a magic state.

Summary. Clifford+ T circuits can be written in terms of $\pi/8$ rotations, $\pi/4$ rotations and measurements. To convert input circuits into a standard form, $\pi/4$ rotations can be commuted to the end of the circuit and absorbed by the final measurements. Thus, any quantum computation can be written as a sequence of $\pi/8$ rotations grouped into layers of mutually commuting rotations. The number of rotations is the T count and the number of layers is the T depth. Each rotation can be performed by consuming a magic state via a Pauli product measurement. These measurements can be implemented in our framework in $1 \odot$.

2 Data blocks

Since Clifford+ T circuits are a sequence of $\pi/8$ rotations, each requiring the consumption of a magic state, it is natural to partition a quantum computer into a set

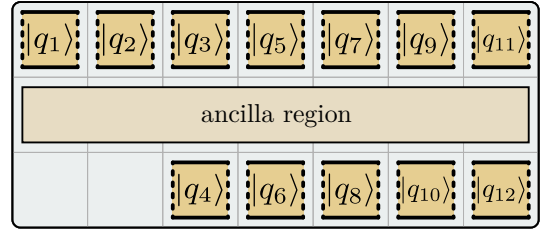


Figure 9: A compact block stores n data qubits in $1.5n + 3$ tiles. The consumption of a magic state can take up to $9 \odot$.

of tiles that are used for magic state distillation (distillation blocks) and a set of tiles that host data qubits and consume magic states via Pauli product measurements (data blocks). In this section, we discuss designs for the latter. In principle, the structure shown in Fig. 8 is a data block, where each qubit is stored in a two-tile patch and magic states can be consumed every $1 \odot$. However, this sort of design uses $3n$ tiles to host n data qubits, which is a relatively large space overhead.

2.1 Compact block

The first design that we discuss uses only $1.5n + 3$ tiles. This compact block is shown in Fig. 9, where each data qubit is stored in a square patch. This lowers the space cost, but restricts the operators that are accessible by Pauli product measurements, as only the Z operator is free to be measured. Using $3 \odot$, patches may also be rotated (see Fig. 11a), such that the X operator becomes accessible instead of the Z operator. The problematic operators are Y operators, which are the reason why the consumption of a magic state can take up to $9 \odot$.

The worst-case scenario is a $\pi/8$ rotation involving an even number of Y operators, such as the one shown in Fig. 10. One possibility to replace Y operators by X or Z operators is via $\pi/4$ rotations, since

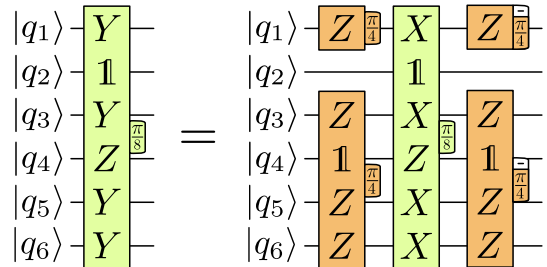


Figure 10: For compact blocks, the worst-case scenario are Pauli product measurements involving an even number of Y operators, e.g., the measurement required for a $(Y \otimes \mathbb{1} \otimes Y \otimes Z \otimes Y \otimes Y)_{\pi/8}$ gate. Such measurements require two explicit $\pi/4$ rotations (left), and two $\pi/4$ rotations that are commuted to the end of the circuit (right).

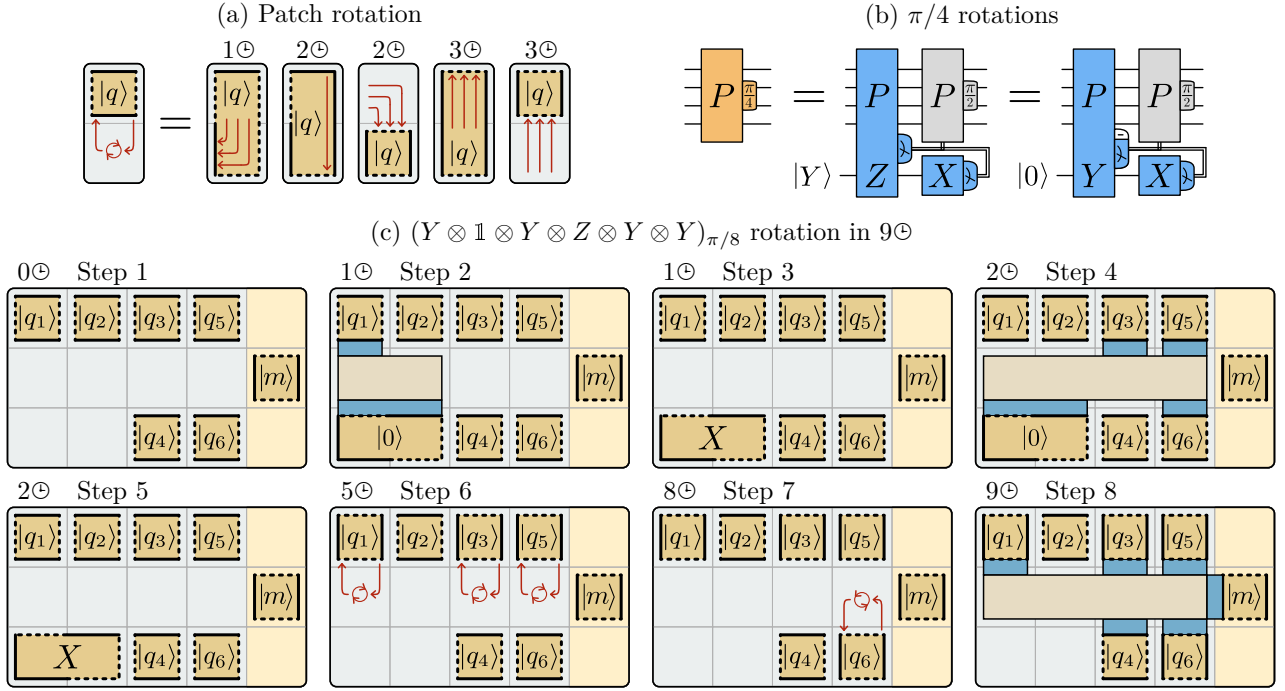


Figure 11: (a) Patches can be rotated in $3\ominus$ to change whether the X or Z operator is adjacent to the compact block's ancilla region. (b) A $P_{\pi/4}$ gate can be performed explicitly via a $P \otimes Y$ measurement with a $|0\rangle$ ancilla qubit. (c) Six-step protocol to perform the rotation of Fig. 10 in a compact block. The magic state is consumed in $9\ominus$, where steps 2-5 are the two $\pi/4$ rotations in Fig. 10, steps 6 and 7 are patch rotations, and step 8 is the Pauli product measurement consuming the magic state.

$Y_{\pi/4} = Z_{\pi/4} X_{\pi/4} Z_{-\pi/4}$. Rotations with an even number of Y 's require two $\pi/4$ rotations, while an odd number of Y 's can be handled by one rotation. Only the left two $\pi/4$ rotations in Fig. 10 need to be performed explicitly. The right two rotations can be commuted to the end of the circuit, changing the subsequent $\pi/8$ rotations. Similarly to a $\pi/8$ rotation, a $P_{\pi/4}$ rotation can be executed using a resource state $|Y\rangle = |0\rangle + i|1\rangle$, as shown in Fig. 11b. However, even though this state is a Pauli eigenstate, it cannot be readily prepared in our framework. Instead, we use a $|0\rangle$ state and Y measurements, such that a $P_{\pi/4}$ rotation is performed by a $P \otimes Y$ measurement between the qubits and the $|0\rangle$ state. Afterwards, the $|0\rangle$ state is measured in X . If the $-P \otimes Y$ and X measurements in Fig. 11b yield different outcomes, a Pauli correction is necessary.

In Fig. 11, we go through the steps necessary to perform the $(Y \otimes 1 \otimes Y \otimes Z \otimes Y \otimes Y)_{\pi/8}$ rotation of Fig. 10. In step 1, we start with a 12-tile data block storing 6 qubits in the blue region. The orange region is not part of the data block, but is part of the adjacent distillation block, i.e., it is the source of the magic states. In steps 2-5, we perform the two $\pi/4$ rotations that are necessary to replace the Y operators with X 's, i.e., the first two $\pi/4$ rotations in the circuit of Fig. 10. In step 6, we first rotate patches in the upper row, and then, in step 7, in

the lower row. Finally, in step 8, we measure the Pauli product involving the magic state.

This general procedure can be used for any $\pi/8$ rotation. First, up to two $\pi/4$ rotations are performed in $2\ominus$. Next, patches in the upper and lower row are rotated, which takes $3\ominus$ per row. Finally, the Pauli product is measured in $1\ominus$, requiring a total of $9\ominus$. While this is very slow compared to Fig. 8, the compact block is a valid choice for small quantum computers where the distillation of a magic state takes longer than $9\ominus$.

2.2 Intermediate block

One possibility to speed up compact blocks is to store all qubits in one row instead of two. This is the intermediate block shown in Fig. 13a, which uses $2n + 4$ tiles to store n qubits. By eliminating one row, all patch rotations can be done simultaneously. In addition, one can save $1\ominus$ by moving all patches to the other side, thereby eliminating the need to move patches back to their row after the rotation. An example is shown in Fig. 12. Suppose we have 5 qubits and need to prepare them for a $Z \otimes X \otimes Z \otimes Z \otimes X$ measurement. The first, third and fourth qubit are moved to the other side, which takes $1\ominus$. Simultaneously, the second and fifth qubit are rotated, which takes $2\ominus$. Therefore, the total

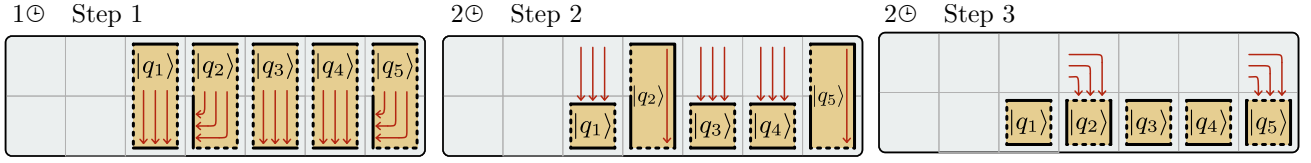


Figure 12: Patch rotations in preparation of a $Z \otimes X \otimes Z \otimes Z \otimes X$ measurement with an intermediate block.

number of time steps to consume a magic state is at most $5\ominus$, where $2\ominus$ are used for up to two $\pi/4$ rotations, $2\ominus$ for the patch rotations, and $1\ominus$ for the Pauli product measurement consuming the magic state.

2.3 Fast block

The disadvantage of square patches is that only one Pauli operator is adjacent to the data block's ancilla region, i.e., available for Pauli product measurements at any given time. Two-tile one-qubit patches as in Fig. 8, on the other hand, allow for the measurement of any Pauli operator, but use two tiles for each qubit. In order to have both compact storage and access to all Pauli operators, we use two-qubit patches for our fast blocks in Fig. 13b. These patches use two tiles to represent two qubits (see Fig. 1), where the first qubit's

Pauli operators are in the left two edges, and the second qubit's operators are in the right two edges. Therefore, the example in Fig. 13b is a fast block that stores 18 qubits.

Since all Pauli operators are accessible, the Pauli product measurement protocol of Fig. 8 can be used to consume a magic state every $1\ominus$. n qubits occupy a square arrangement of tiles with a side length of $\sqrt{n/2} + 1$, i.e., a total of $2n + \sqrt{8n} + 1$ tiles. Even if $\sqrt{n/2}$ is not integer, one should keep the block as square-shaped as possible by picking the closest integer as a side length and shortening the last column. While the fast block uses more tiles compared to the compact and intermediate blocks, it has a lower space-time cost, making it more favorable for large quantum computers for which the distillation of a magic state takes less than $5\ominus$.

Note that if undistilled magic states are sufficient, then any data block can already be used as a full quantum computer. A proof-of-principle two-qubit device in the spirit of Ref. [31] that constitutes a universal two-qubit quantum computer with undistilled magic states and can demonstrate all the operations that are used in our framework can be realized with six tiles, as shown in Appendix C. This proof-of-principle device uses $(3d - 1) \cdot 2d$ physical data qubits, i.e., 48, 140, or 280 data qubits for distances $d = 3, 5$ or 7 . If ancilla qubits are used for stabilizer measurements, the number of physical qubits roughly doubles, but it is still within reach of near-term devices.

Summary. Data blocks store the data qubits of the computation and consume magic states. Compact blocks use $1.5n + 3$ tiles for n qubits and require up to $9\ominus$ to consume a magic state. Intermediate blocks use $2n + 4$ tiles and take up to $5\ominus$ per magic state. Fast blocks use $2n + \sqrt{8n} + 1$ tiles and take $1\ominus$ per magic state. Data blocks need to be combined with distillation blocks for large-scale quantum computation.

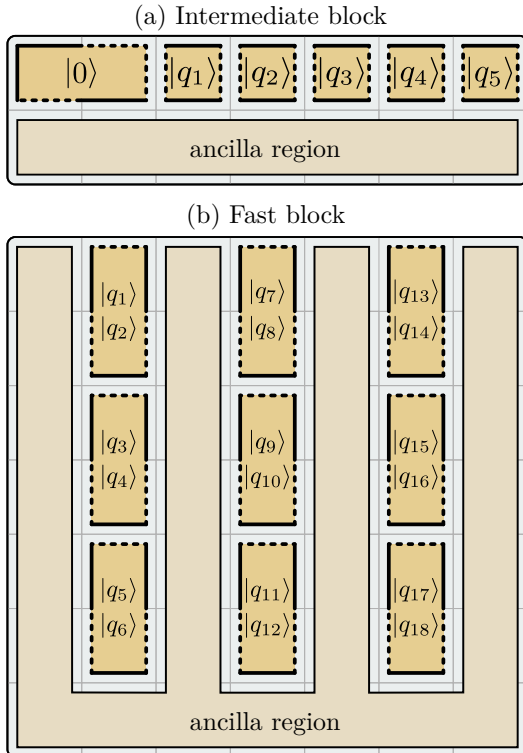


Figure 13: (a) Intermediate blocks store n data qubits in $2.5n + 4$ tiles and require up to $5\ominus$ per magic state. (b) Fast blocks use $2n + \sqrt{8n} + 1$ tiles and require $1\ominus$ per magic state.

3 Distillation blocks

In this section, we discuss designs of tile blocks that are used for magic state distillation. This is necessary, because with surface codes, the initialization of non-Pauli eigenstates is prone to errors, which means that

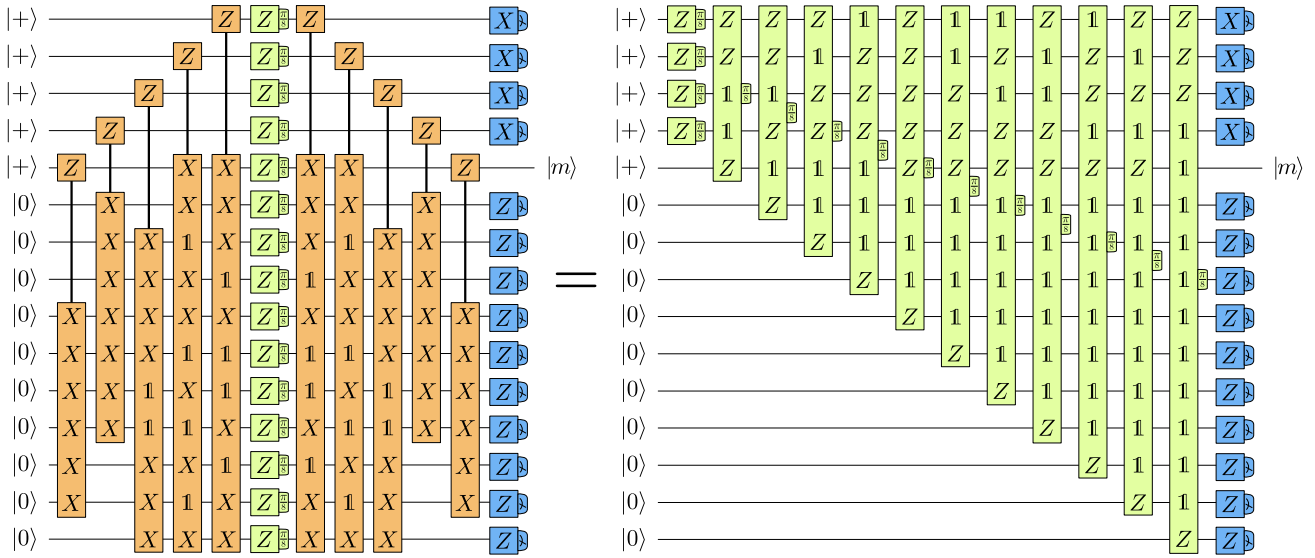


Figure 14: Encode- T -decode circuit of the 15-to-1 distillation protocol. The multi-target CNOTs (orange) can be commuted past the T gates, such that they cancel and leave 15 Z -type Pauli product rotations.

$\pi/8$ rotations performed using these states may lead to errors. In order to decrease the probability of such an error, magic state distillation [16] is used to convert many low-fidelity magic states into fewer higher-fidelity states. This requires only Clifford gates (i.e., Pauli product measurements), so, in principle, any of the data blocks discussed in the previous section can be used for this purpose. However, magic state distillation is repeated extremely often for large-scale quantum computation, so it is worth optimizing these protocols.

Here, we discuss a general procedure that can be applied to any distillation protocol based on an error-correcting code with transversal T gates, such as punctured Reed-Muller codes [16, 17] or block codes [18–20]. To show the general structure of such a protocol, we go through the example of 15-to-1 distillation [16], i.e., a protocol that uses 15 faulty magic states to distill a single higher-fidelity state.

3.1 15-to-1 distillation

The 15-to-1 protocol is based on a quantum error-correcting code that uses 15 qubits to encode a single logical qubit with code distance 3. The reason why this can be used for magic state distillation is that, for this code, a physical T gate on every physical qubit corresponds to a logical T gate (actually T^\dagger) on the encoded qubit, which is called a transversal T gate. The general structure of a distillation circuit based on a code with transversal T gates is shown in Fig. 14 for the example of 15-to-1. It consists of four parts: an encoding circuit, transversal T gates, decoding and measurement.

The circuit begins with 5 qubits initialized in the $|+\rangle$ state and 10 qubits in the $|0\rangle$ state. Qubits 1-4, 5 and 6-15 are associated with the four X stabilizers, the logical X operator, and the ten Z stabilizers of the code. The first five operations are multi-target CNOTs that correspond to the code’s encoding circuit. They map the X Pauli operators of qubits 1-4 onto the code’s X stabilizers, the X Pauli of qubit 5 onto the logical X operator and the Z operators of qubits 6-15 onto the code’s Z stabilizers. Because we start out with $+1$ -eigenstates of X and Z , this circuit prepares the simultaneous stabilizer eigenstate corresponding to the logical $|+\rangle_L$ state. Next, a transversal T gate is applied, transforming the logical state to $T_L |+\rangle_L$ (actually to $T_L^\dagger |+\rangle_L$). Note that the 15 $Z_{\pi/8}$ rotations are potentially faulty. Finally, the encoding circuit is reverted, shifting the logical qubit information back into qubit 5, and the information about the X and Z stabilizers into qubits 1-4 and 6-15. If no errors occurred, qubit 5 is now a magic state $T|+\rangle$ (actually $T^\dagger |+\rangle$). In order to detect whether any of the 15 $\pi/8$ rotations were affected by an error, qubits 1-4 and 6-15 are measured in the X and Z basis, respectively, effectively measuring the stabilizers of the code. Since the code distance is 3, up to two errors can be detected, which will yield a -1 measurement outcome on some stabilizers. If any error is detected, all qubits are discarded and the distillation protocol is restarted. This way, if the error probability of each of the 15 T gates is p , the error probability of the output state is reduced to $35p^3$ to leading order. In other words, this protocol takes 15 magic states with error probability p , and outputs a single magic state with an error of $35p^3$.

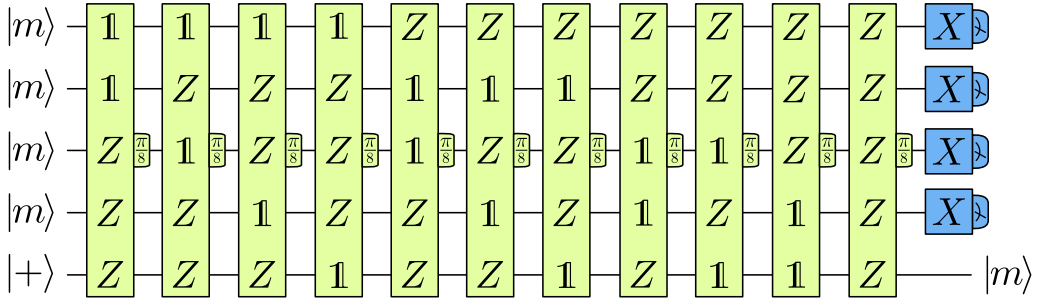


Figure 15: 15-to-1 distillation circuit that uses 5 qubits and 11 $\pi/8$ rotations.

Simplifying the circuit. Using the commutation rules of Fig. 4b, we can commute the first set of multi-target CNOTs to the right. This maps the $Z_{\pi/8}$ rotations onto Z -product $\pi/8$ rotations. Since controlled-Pauli gates satisfy $C(P_1, P_2) = C(P_1, P_2)^\dagger$, the multi-target CNOTs of the encoding circuit precisely cancel the multi-target CNOTs of the decoding circuit, leaving a circuit of 15 Z -type $\pi/8$ rotations in Fig. 14.

Note that qubits 6-15 in this circuit are entirely redundant. They are initialized in a Z eigenstate, are then part of a Z -type rotation, and are finally measured in the Z basis, trivially yielding the outcome $+1$. Since they serve no purpose, they can simply be removed to yield the five-qubit circuit in Fig. 15, where we have absorbed the single-qubit $\pi/8$ rotations into the initial $|+\rangle$ states and rearranged the remaining 11 rotations.

This kind of circuit simplification is equivalent to the space-time trade-offs mentioned in Ref. [17] and can be applied to any protocol that is based on a code with transversal T gates. In general, a code with m_x X stabilizers that uses n qubits to encode k logical qubits yields a circuit of $n - m_x$ $\pi/8$ rotations on $m_x + k$ qubits. Each of the $m_x + k$ qubits are either associated with an X stabilizer or one of the k logical qubits. For each of the n qubits of the code, the circuit contains one $\pi/8$ rotation with an axis that has a Z on each stabilizer or logical X operator that this qubit is part of. In order to more easily determine the $n - m_x$ rotations, it is useful to write down an $n \times (m_x + k)$ matrix that shows the X stabilizers and logical X operators of the code. For 15-to-1, such a matrix could look like this:

$$M_{15\text{-to-1}} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \quad (1)$$

Each of the first four rows describes one of the four X stabilizers of the code, where 0 stands for $\mathbb{1}$ and 1 stands for X . For instance, the first row indicates that

the first X stabilizer of this 15-qubit code is $\mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes X \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes X \otimes X \otimes X \otimes X \otimes X \otimes X \otimes X$. The rows below the horizontal bar – in this case the last row – show the logical X operators of the code. The circuit in Fig. 15 is then obtained by placing a $|+\rangle$ state for each row and a $\pi/8$ rotation for each column, with the axis of rotation determined by the indices in the column – a $\mathbb{1}$ for each 0 and a Z for each 1. Note that, in Fig. 15, the first four rotations (columns) of Eq. (1) are absorbed by the initial states.

3.2 Triorthogonal codes

The aforementioned circuit translation can be applied to any code with transversal T gates. One particularly versatile and simple scheme to generate such codes is based on triorthogonal matrices [17, 18], which we briefly review in this section. The first step is to write down a triorthogonal matrix G , such as

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}. \quad (2)$$

Triorthogonality refers to three criteria: *i*) The number of 1s in each row is a multiple of 8. *ii*) For each pair of rows, the number of entries where both rows have a 1 is a multiple of 4. *iii*) For each set of three rows, the number of entries where all three rows have a 1 is a multiple of 2. In other words,

$$\begin{aligned} \forall a : \sum_i G_{a,i} &= 0 \pmod{8} \\ \forall a, b : \sum_i G_{a,i} G_{b,i} &= 0 \pmod{4} \\ \forall a, b, c : \sum_i G_{a,i} G_{b,i} G_{c,i} &= 0 \pmod{2} \end{aligned} \quad (3)$$

A general procedure based on classical Reed-Muller codes to obtain such matrices is described in Ref. [17].

After obtaining a triorthogonal matrix, such as the one in Eq. (2), the second step is to put it in a row

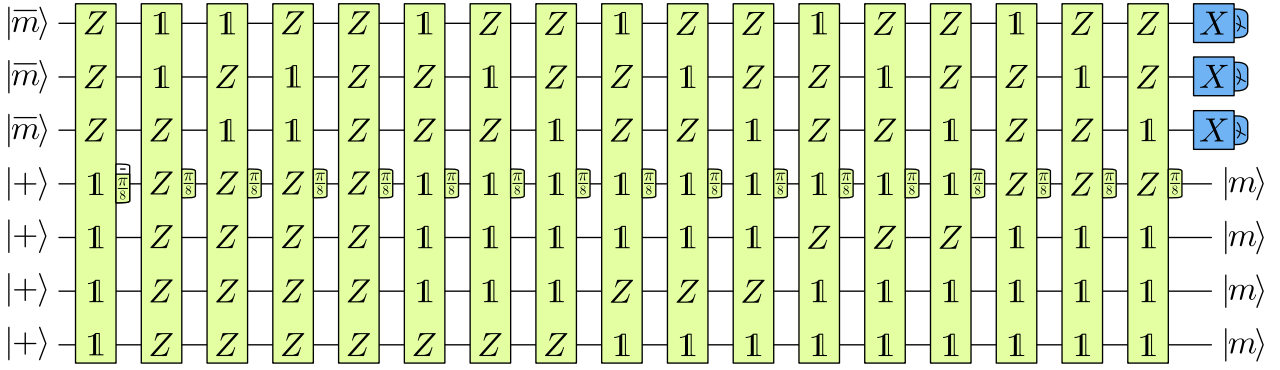


Figure 16: 20-to-4 distillation circuit that uses 7 qubits and 17 $\pi/8$ rotations.

echelon form by Gaussian elimination

$$\tilde{G} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}. \quad (4)$$

The last step is to remove one of the columns that contains a single 1, i.e., one of the first five columns, which is also called puncturing.² Puncturing an $a \times b$ triorthogonal matrix k times yields a code encoding k logical qubits with $m_x = b - k$ and $n = a - k$. The rows of the matrix after puncturing that contain an even number of 1s describe X stabilizers, whereas the rows with an odd number of 1s describe X logical operators. In terms of distillation protocols, a code described by such a matrix can be used for n -to- k distillation. Indeed, if we puncture the matrix in Eq. (4) once by removing the first column, we retrieve the 15-to-1 protocol of Eq. (1). We can also puncture it twice by removing the first two columns. This yields the matrix

$$M_{14\text{-to-2}} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}, \quad (5)$$

which describes a 14-to-2 protocol. The corresponding circuit can be simply read off from this matrix. It is almost identical to the 15-to-1 protocol of Fig. 15, except that the fourth qubit is initialized in the $|+\rangle$ state and is not measured at the end of the circuit, but instead outputs a second magic state. However, because the code of 14-to-2 has a code distance of 2, the output error probability is higher, namely $7p^2$ [18]. Puncturing the matrix \tilde{G} any further would yield codes with a

²Even though this is commonly called puncturing, it would be perhaps more accurate to refer to this process as *shortening* (see, e.g., Ref. [32]), as was pointed out to me by a referee.

distance lower than 2, precluding them from detecting errors and improving the quality of magic states. In fact, the minimum number of qubits in triorthogonal codes was shown to be 14 [33].

Semi-triorthogonal codes. There are also codes that are based on “semi-triorthogonal” matrices, where all three conditions of Eq. (3) are only satisfied modulo 2. One example is the matrix

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \quad (6)$$

When this matrix is punctured four times, it yields a code that can be used for a 20-to-4 protocol. A scheme to generate such matrices for $3k+8$ -to- k distillation is shown in Ref. [18]. For the case of the 20-to-4 protocol, the matrix that describes the code

$$M_{20\text{-to-4}} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad (7)$$

can be straightforwardly translated into the circuit in Fig. 16. While semi-triorthogonal codes can be used the same way for distillation as properly triorthogonal codes, their caveat is that a Clifford correction may be required. This correction can be obtained by adding columns to the semi-triorthogonal matrix until it becomes properly triorthogonal, e.g., by adding the

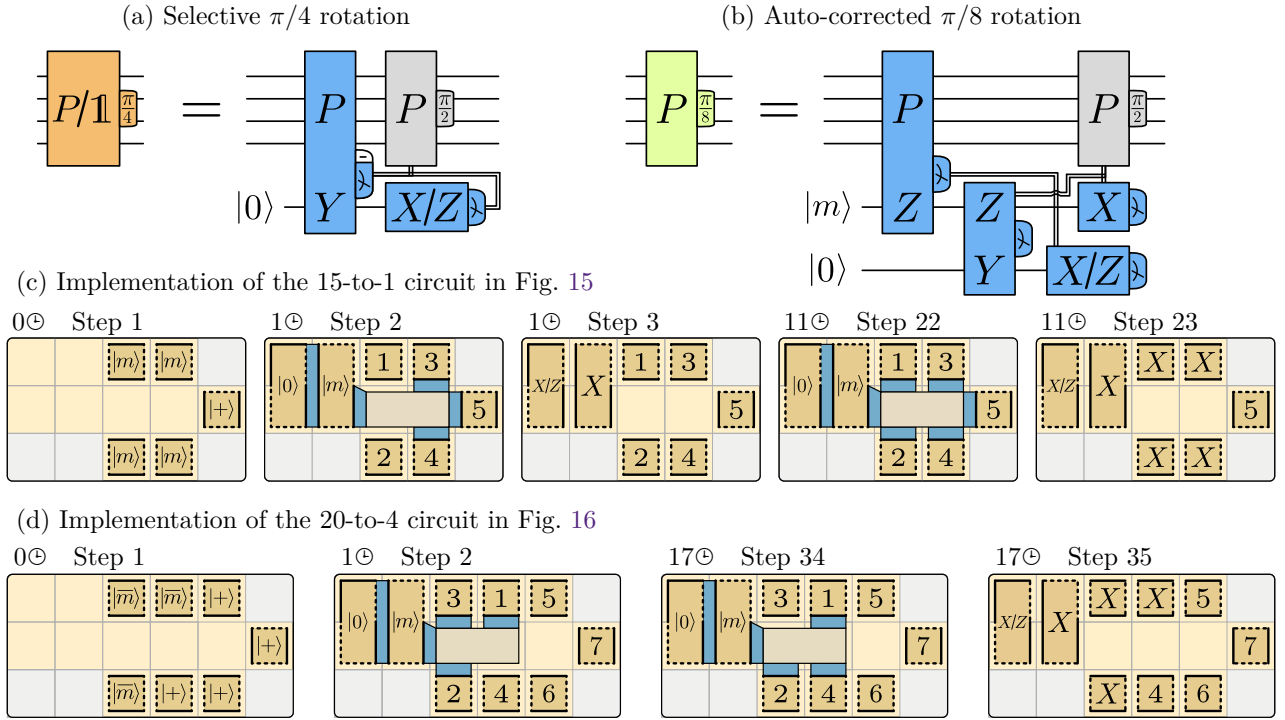


Figure 17: Implementation of the 15-to-1 and 20-to-4 distillation protocols in our framework. Each time step in (c) and (d) corresponds to an auto-corrected $\pi/8$ rotation (b), which in turn is based on selective $\pi/4$ rotations (a).

columns of the matrix

$$M_{\text{Clifford correction}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (8)$$

to the matrix of Eq. (7). Since the additional columns come in pairs, this Clifford correction always consists of Z -type $\pi/4$ rotations [18].

In this case, the correction consists of four $\pi/4$ rotations on the first three qubits, effectively changing the first $(Z \otimes Z \otimes Z)_{\pi/8}$ rotation to a $(Z \otimes Z \otimes Z)_{-\pi/8}$ rotation, and the initial magic states to $|\overline{m}\rangle = |0\rangle + e^{-i\pi/4}|1\rangle$ states. The probability of any of the four output states being affected by an error is $22p^2$. When treating this output error rate as $5.5p^2$ per magic state, one should take into account that, for multiple output states, errors can be correlated. Note that $3k+8$ -to- k protocols can be modified to $3k+4$ -to- k [33–35].

3.3 Surface-code implementation

Having outlined the general structure of distillation protocols, we now discuss their implementation with sur-

face codes. Distillation protocols are particularly simple quantum circuits, since they exclusively consist of Z -type $\pi/8$ rotations. Therefore, we can use a construction similar to the compact data block, and still only require $1\ominus$ per rotation.

Because distillation circuits are relatively short, it is useful to avoid the Clifford corrections of Fig. 7 that may be required with 50% probability after a magic state is consumed. These corrections slow down the protocol, because they change the final X measurements to Pauli product measurements. Instead, we use a circuit which consumes a magic state and automatically performs the Clifford correction. It is based on the selective $\pi/4$ rotation circuit in Fig. 17a. To perform a $P_{\pi/4}$ rotation according to the circuit in Fig. 11b, a $|0\rangle$ state is initialized and $P \otimes Y$ is measured, which takes $1\ominus$. However, the $\pi/4$ rotation is only performed if the $|0\rangle$ qubit is measured in X afterwards. If, instead, it is measured in Z , the qubit is simply discarded without performing any operation. In other words, the choice of measurement basis determines whether a $P_{\pi/4}$ or a $\mathbb{1}$ operation is performed. This can be used to construct the circuit in Fig. 17b. Here, the first step to perform a $P_{\pi/8}$ gate is to measure $P \otimes Z$ between the qubits and a magic state $|m\rangle$, and $Z \otimes Y$ between $|m\rangle$ and $|0\rangle$. These two measurements commute and can be performed simultaneously. If the outcome of the first measurement

is +1, no Clifford correction is required and $|0\rangle$ is read out in Z . If the outcome is -1, $|0\rangle$ is measured in X , yielding the required Clifford correction.

This can be used to implement the 15-to-1 protocol of Fig. 15 in $11\ominus$ using 11 tiles, as shown in Fig. 17c. Four qubits are initialized in $|m\rangle$, and a fifth in $|+\rangle$. A 2×2 block of tiles to the left is reserved for the $|m\rangle$ and $|0\rangle$ qubits of the auto-corrected $\pi/8$ rotations. Two additional tiles are used for the ancilla of the multi-patch measurement. In step 2, the first $\pi/8$ rotation $(\mathbb{1} \otimes \mathbb{1} \otimes Z \otimes Z \otimes Z)_{\pi/8}$ is performed. Depending on the measurement outcome of step 2, the $|0\rangle$ ancilla is read out in the X or Z basis. This is repeated 11 times, once for each of the 11 rotations in Fig. 15. Finally, in step 23, qubits 1-4 are measured in X . If all four outcomes are +1, the distillation protocol yields a distilled magic state in tile 5. Since 11 tiles are used for $11\ominus$, the space-time cost is $121d^3$ in terms of (physical data qubits)·(code cycles) to leading order. Similarly, the 20-to-4 protocol of Fig. 16 is implemented in Fig. 17d using 14 tiles for $17\ominus$, i.e., with a leading-order space-time cost of $238d^3$.

Caveat. Even though our leading-order estimate of the time cost of $11d$ code cycles for 15-to-1 or $17d$ code cycles for 20-to-4 is correct, the full time cost also contains contributions that do not scale with d . The two processes that may require special care in the magic state distillation protocol are state injection and classical processing. Every $1\ominus$ requires the initialization of a magic state and a short classical computation to determine whether the $|0\rangle$ state needs to be measured in X or Z . While neither of these processes scales with d , they can slow down the distillation protocol, depending on the injection scheme and the control hardware that is used. This slowdown can be avoided by using additional 2×2 blocks of $|0\rangle$ - $|m\rangle$ pairs, as shown in Fig. 18 for 15-to-1 distillation with one additional block. Here, the left and right block can be used in an alternating fashion, i.e., the left block for rotations 1, 3, 5, ... and the right block for rotations 2, 4, 6, ... While one block is being used for a rotation, the other one can be used to prepare a new magic state and to process the measurement outcomes of the previous rotation.

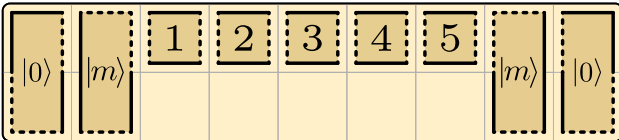


Figure 18: Two 2×2 ancilla blocks can be used to prevent state injection and classical processing from slowing down the 15-to-1 protocol.

General space-time cost. The scheme of Fig. 17 can be used to implement any protocol based on a triorthogonal code. For an n -qubit code with k logical qubits and m_x X stabilizers, the protocol uses $1.5(m_x + k) + 4$ tiles for $(n - m_x) \ominus$. In this time, it distills k magic states with a success probability of $\sim(1 - p)^n$, since any error will result in failure. Therefore, such a protocol distills k magic state on average every $(n - m_x)/(1 - p)^n$ time steps. Thus, the space-time cost per magic state is

$$\text{cost}(n, m_x, k, p, d) = \frac{[1.5(m_x + k) + 4](n - m_x)}{k(1 - p)^n} d^3. \quad (9)$$

In order to minimize the space-time cost for distillation in our framework, one should pick a distillation protocol that minimizes this quantity for a given input and target error rate.

3.4 Benchmarking

We can use the previously described 15-to-1 and 20-to-4 schemes to benchmark our implementations. In Ref. [36], these schemes were implemented with lattice surgery and their cost compared to implementations based on braiding of hole defects. In addition, the 7-to-1 scheme was considered, which is a scheme to distill $|Y\rangle$ states. The distillation of these states is not necessary in our framework, but for benchmarking purposes we show the 7-to-1 protocol in Appendix D. It can be implemented using 7 tiles for $4\ominus$, i.e., with a space-time cost of $28d^3$.

We summarize the leading-order space-time costs of the three protocols in Table 1. The comparison shows drastic reductions in space-time cost compared to schemes based on braiding of hole defects and compared to other approaches to optimizing lattice surgery. Compared to the braiding-based scheme, the space-time cost of 7-to-1, 15-to-1 and 20-to-4 is reduced by 60%, 84% and 90%, respectively.

	7-to-1	15-to-1	20-to-4
Hole braiding [20, 37]	$70d^3$	$750d^3$	$2344d^3$
Lattice surgery [36]	$140d^3$	$540d^3$	$1134d^3$
Our framework	$28d^3$	$121d^3$	$238d^3$

Table 1: Comparison of the leading-order space-time cost of 7-to-1, 15-to-1 and 20-to-4 with defect-based schemes, optimized lattice surgery in Ref. [36] and our schemes. The space-time cost is in terms of (physical data qubits)·(code cycles).

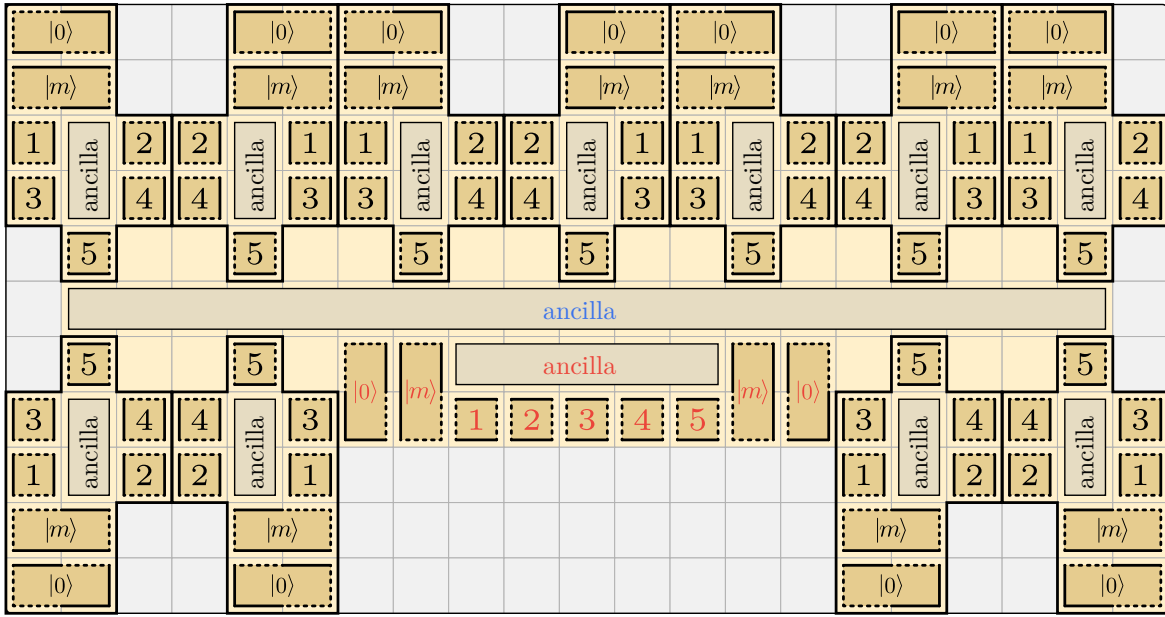


Figure 19: 176-tile block that can be used for 225-to-1 distillation. The qubits highlighted in red are used for the second level of the distillation protocol. The blue ancilla is used to move level-1 magic states into the two $|m\rangle$ - $|0\rangle$ blocks of the level-2 distillation.

3.5 Higher-fidelity protocols

So far, we have only explicitly discussed protocols that reduce the input error to $\sim p^2$ or $\sim p^3$. There are two strategies to obtain protocols with a higher output fidelity: concatenation and higher-distance codes.

Concatenation. In the 15-to-1 protocol, we use 15 undistilled magic states to obtain a distilled magic state with an error rate of $35p^3$. If we perform the same protocol, but use 15 distilled magic states from previous 15-to-1 protocols as inputs, the output state will have an error rate of $35(35p^3)^3 = 1500625p^9$. This corresponds to a 225-to-1 protocol obtained from the concatenation of two 15-to-1 protocols. It is also possible to concatenate protocols that are not identical. Strategies to combine high-yield and low-yield protocols are discussed in Ref. [18].

In Fig. 19, we show an unoptimized block that can be used for 225-to-1 distillation. It consists of 11 15-to-1 blocks that are used for the first level of distillation. Since each of these 11 blocks takes $11\ominus$ to finish, they can be operated such that exactly one of these blocks finishes in every time step. Therefore, in every time step, one first-level magic state can be used for second-level distillation by moving it into one of the two level-2 $|m\rangle$ - $|0\rangle$ blocks via the blue ancilla. The qubits that are used for the second level are highlighted in red. Note that since, for the second level, the single-qubit $\pi/8$ rotations require distilled magic states, the 15-to-1 protocol of Fig. 15 requires 15 rotations instead of

just 11. Therefore, the entire protocol finishes in $15\ominus$ using 176 tiles with a total space-time cost of $2640d^3$. It should be noted that, since lower-level distillation blocks produce magic states with low fidelity, there is no benefit in using the full code distance to produce these states. The space-time cost of concatenated protocols can be reduced significantly by running the lower-level distillation blocks at a reduced code distance (see, e.g., Refs. [12, 38]), using smaller patches and fewer code cycles. The exact code distance that should be used depends on the protocol and the desired output fidelity.

Higher-distance codes. Alternatively, we can use a code that produces higher-fidelity states. In Ref. [17], several protocols based on punctured Reed-Muller codes are discussed. One of these protocols is a 116-to-12 protocol based on a code with $n = 116$, $k = 12$ and $m_x = 17$. It yields 12 magic states which each have an error rate of $41.25p^4$. According to Eq. (9), this protocol can be implemented using 44 tiles for $99\ominus$ with a space-time cost of $363d^3$ per output state and a success probability of $(1-p)^{116}$. For protocols with a high space cost such as 116-to-12, the space-time cost can be slightly reduced by introducing additional ancilla space, such that two operations can be performed simultaneously. One possible configuration is shown in Fig. 20. This increases the space cost to 81 tiles, but reduces the time cost to $50\ominus$, with a total space-time cost of $337.5d^3$ per output state.

Output-to-input ratio is not everything. A popular figure of merit when comparing n -to- k distillation

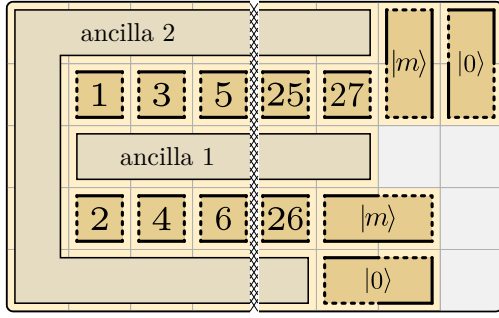


Figure 20: 81-tile block that can be used for the 116-to-12 protocol. Here, two $\pi/8$ rotations can be performed at the same time, where one rotation uses the ancilla space denoted as *ancilla 1*, and the other one uses *ancilla 2*.

protocols is the ratio k/n . One of the protocols in Ref. [17] is a 912-to-112 protocol with $n = 912$, $k = 112$ and $m_x = 64$, which yields 112 output state, each with an error rate of $10.63p^6$. While the output fidelity is not as high as for 225-to-1, the output-to-input ratio is much higher. For $p = 10^{-3}$, the output fidelity of 225-to-1 is $\sim 1.5 \times 10^{-21}$, while it is only $\sim 10^{-17}$ for 912-to-112. Therefore, if output-to-input ratio were a good figure of merit, we would expect the 912-to-112 protocol to be considerably less costly compared to 225-to-1. If we use an implementation in the spirit of Fig. 20, the space cost is roughly $2.5(m_x + k)$ tiles and the protocol takes $(n - m_x)/2$ time steps. Thus, 912-to-112 uses 440 tiles for 424 \odot . This would put the space-time cost per state at $1665d^3$, which is indeed lower than that of 225-to-1. However, the success probability of 912-to-112 for $p = 10^{-3}$ is only at $\sim 40\%$, which more than doubles the actual space-time cost. On the other hand, the space-time cost of 225-to-1 is barely affected by the success probability, as each of the level-1 15-to-1 blocks finishes with 98.5% success probability. This means that, with 1.5% probability, a time step of 225-to-1 is skipped, since the necessary level-1 state is missing. This only increases the space-time cost from 2640^3 to $2680d^3$. Even without further decreasing the space-time cost of 225-to-1 by reducing the code distance of the level-1 distillation blocks, this indicates that the output-to-input ratio is not a good figure of merit in our framework.

Summary. The class of magic state distillation protocols that are based on an n -qubit error-correcting code with m_x X stabilizers and k logical qubits can be implemented using $1.5(m_x + k) + 4$ tiles and $n - m_x$ time steps. Such protocols output k magic states with a success probability of $(1 - p)^n$. Therefore, if the input fidelity and desired output fidelity are known, the distillation protocol should minimize the cost function given in Eq. (9).

4 Trade-offs limited by T count

Having discussed data blocks and distillation blocks in the previous two sections, we are now ready to piece them together to a full quantum computer. In order to illustrate the steps that are necessary to calculate the space and time cost of a computation and to trade off space against time, we consider an example computation with a T count of 10^8 and a T depth of 10^6 . We consider two different scenarios: an error rate of $p = 10^{-3}$ and an error rate of $p = 10^{-4}$. The error rate determines how many physical qubits are required per logical qubit and which distillation protocol should be used. It is only a meaningful number, if we specify an error model for the physical qubits and undistilled magic states. We will assume circuit-level noise for the physical qubits, i.e., faulty qubits, gates and measurements. The error model for undistilled magic states depends on the specific state-injection protocol. We will assume that raw magic states are affected by random Pauli errors with probability p . To calculate concrete numbers, we assume that the quantum computer can perform a code cycle every $1 \mu\text{s}$. We want to perform the 10^8 - T -gate computation in a way that the probability of any one of the T gates being affected by an error stays below 1%. In addition, we require that the probability of an error affecting any of the logical qubits encoded in surface-code patches stays below 1%. This results in a 2% chance that the quantum computation will yield a wrong result. In order to exponentially increase the precision of the computation, it can be repeated multiple times or run in parallel on multiple quantum computers.

4.1 Step 1: Determine distillation protocol

The first step is to determine which distillation protocol is sufficient for the computation. In order to stay below 1% error probability with 10^8 T gates, each magic state needs to have an error rate below 10^{-10} . For $p = 10^{-4}$, the 15-to-1 protocol is sufficient, since it yields an output error rate of $35p^3 = 3.5 \cdot 10^{-11}$. For $p = 10^{-3}$, 15-to-1 is not enough. On the other hand, two levels of 15-to-1, i.e., 225-to-1, yield magic states with an error rate of $1.5 \cdot 10^{-21}$, which is many orders of magnitude above what is required. A less costly protocol is 116-to-12, which yields output states with an error rate of $41.25p^4 = 4.125 \cdot 10^{-11}$, which suffices for our purposes.

4.2 Step 2: Construct a minimal setup

In order to determine the necessary code distance, we first construct a minimal setup, i.e., a configuration of tiles that can be used for the computation and uses as little space as possible. The reason why this is useful

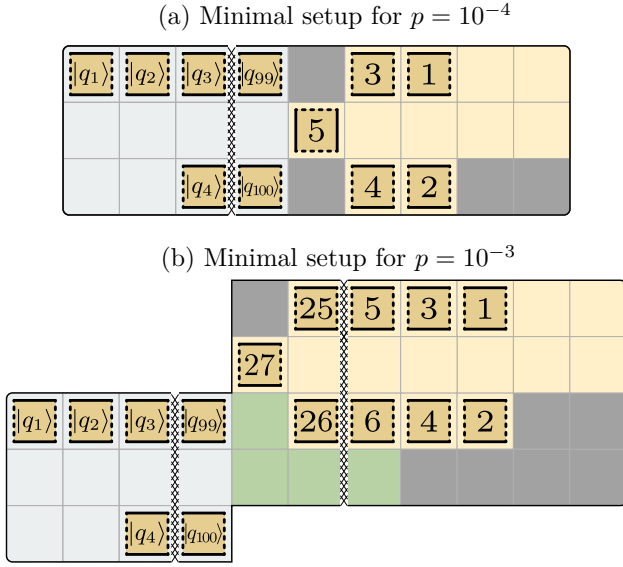


Figure 21: Minimal setups using compact data blocks for $p = 10^{-4}$ (with 15-to-1 distillation) and $p = 10^{-3}$ (with 116-to-12 distillation). Blue tiles are data block tiles, orange tiles are distillation block tiles, green tiles are used for magic state storage and gray tiles are unused tiles.

to determine the code distance is that the initial space-time trade-offs that we discuss significantly improve the overall space-time cost. Therefore, the minimal setup can be used to comfortably upper-bound the required code distance.

For $p = 10^{-4}$, a minimal setup consists of a compact data block and a 15-to-1 distillation block, see Fig. 21a. The compact block stores 100 qubits in 153 tiles and requires up to $9\ominus$ to consume a magic state. The 15-to-1 distillation block uses 11 tiles and outputs a magic state every $11\ominus$ with 99.9% success. To ensure that the tile of the distillation block that is occupied by qubit 5 is not blocked during the first time step of the distillation protocol, the first $\pi/8$ rotation of the protocol should be chosen such that it does not involve qubit 5, e.g., the fourth rotation of Fig. 15. In total, this minimal setup uses 164 tiles and performs a T gate every $11\ominus$, i.e., finishes the computation in $11 \cdot 10^8$ time steps.

For $p = 10^{-3}$, a minimal setup consists of a compact data block and a 116-to-12 distillation block, as shown in Fig. 21b. For the minimal setup, we do not use the larger and faster distillation block shown in Fig. 20, but instead a block in the spirit of the 15-to-1 block. This 116-to-12 distillation block uses 44 tiles and distills 12 magic states in $99\ominus$ with 89% success probability, i.e., on average one state every $9.27\ominus$. Because this distillation protocol outputs magic states in bursts, i.e., 12 at the same time, these states need to be stored before being consumed. Therefore, we introduce additional

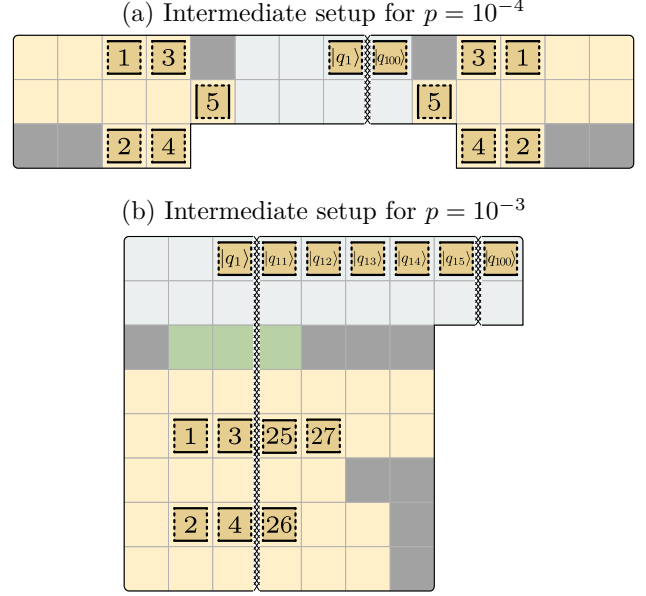


Figure 22: Intermediate setups using intermediate data blocks and two 15-to-1 distillation blocks for $p = 10^{-4}$ or one compact 116-to-12 distillation block for $p = 10^{-3}$.

storage tiles (green tiles in Fig. 21b). Here, we choose the 12 output states to be qubits 6, 8, 10, \dots , 26 and 27. In the last step of the protocol these states are moved into the green space, where they are consumed by the data block one after the other. This minimal setup uses 153 tiles for the data block, 44 tiles for the distillation block and 13 tiles for storage. In total, it uses 210 tiles and finishes the computation in $9.27 \cdot 10^8$ time steps.

4.3 Step 3: Determine code distance

Since each tile corresponds to $d \times d$ physical data qubits and each time step corresponds to d code cycles, 164 encoded logical qubits need to survive for $(11 \cdot 10^8)d$ code cycles for the minimal setup with $p = 10^{-4}$. The probability of a single logical error on any of these 164 qubits needs to stay below 1% at the end of the computation. The logical error rate per logical qubit per code cycle can be approximated [12] as

$$p_L(p, d) = 0.1(100p)^{(d+1)/2} \quad (10)$$

for circuit-level noise. Therefore, the condition to determine the required code distance is

$$164 \cdot 11 \cdot 10^8 \cdot d \cdot p_L(10^{-4}, d) < 0.01. \quad (11)$$

For distance $d = 11$, the final error probability is at 19.8%. Therefore, distance $d = 13$ is sufficient, with a final error probability of 0.2%. The number of physical qubits used in the minimal setup can be calculated

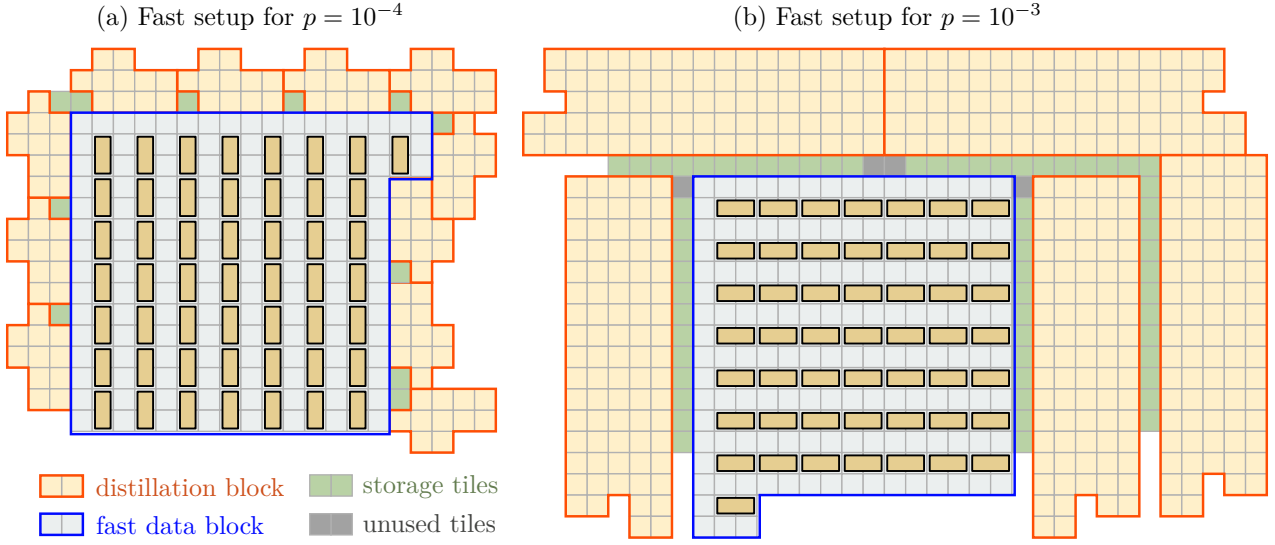


Figure 23: Fast setups using fast data blocks and 11 15-to-1 distillation blocks for $p = 10^{-4}$ or 5 116-to-12 distillation block for $p = 10^{-3}$.

as the number of tiles multiplied by $2d^2$, taking measurement qubits into account. The minimal setup for $p = 10^{-4}$ uses $164 \cdot 2 \cdot 13^2 \approx 55,400$ physical qubits and finishes the computation in $13 \cdot 11 \cdot 10^8$ code cycles. With $1 \mu\text{s}$ per code cycle, this amounts to roughly 4 hours.

For $p = 10^{-3}$, the condition changes to

$$210 \cdot 9.27 \cdot 10^8 \times d \cdot p_L(10^{-3}, d) < 0.01, \quad (12)$$

which is satisfied for $d = 27$ with a final error probability of 0.5%. The final error probability for $d = 25$ is at 4.9%. Thus, the minimal setup uses $210 \cdot 2 \cdot 27^2 \approx 306,000$ physical qubits and finishes the computation in $27 \cdot 9.27 \cdot 10^8$ code cycles, which amounts to roughly 7 hours. Note that, in principle, a success probability of less than 50% would be sufficient to reach arbitrary precisions by repeating computations or running them in parallel. This means that the code distances that we consider may be higher than what is necessary.

4.4 Step 4: Add distillation blocks

Only a small fraction of the tiles of the minimal setup is used for magic state distillation, i.e., 6.7% for $p = 10^{-4}$ and 21% for $p = 10^{-3}$. On the other hand, adding one additional distillation block doubles the rate of magic state production, potentially doubling the speed of computation. Therefore, in order to speed up the computation and decrease the space-time cost, we add additional distillation blocks to our setup.

For $p = 10^{-4}$, adding one more distillation block reduces the time that it takes to distill a magic state to $5.5\ominus$ per state. However, the compact block can only consume magic states at $9\ominus$ per state. In order to

avoid this bottleneck, we can use the intermediate data block instead, which occupies 204 tiles, but consumes one magic state every $5\ominus$. With 22 tiles for distillation (see Fig. 22), this setup uses 226 tiles and finishes the computation after $5.5 \cdot 10^8$ time steps. This increases the number of qubits to 76,400, but reduces the computational time to 2 hours.

For $p = 10^{-3}$, the addition of a distillation block reduces the distillation time to $4.64\ominus$. At this point, one should switch to the more efficient 116-to-12 block of Fig. 20, which uses 81 tiles and distills a magic state on average every $4.68\ominus$. The intermediate data block cannot keep up with this distillation rate, but we can still use it to consume one magic state every $5\ominus$ instead of $4.68\ominus$. Such a configuration uses 228 data tiles, 81 distillation tiles and 13 storage tiles, i.e., a total of 322 tiles corresponding to approximately 469,000 physical qubits. The computational time reduces to $5 \cdot 10^8$ time steps, i.e., 3.75 hours. Note that in Fig. 22b, the 12 output states of the 116-to-12 protocol should be chosen as 1, 3, 5, \dots , 25. They can be moved into the green storage space in the last step of the protocol, since the space denoted as *ancilla 2* in Fig. 20 is not being used in the last step.

Trade-offs down to $1\ominus$ per T gate. Adding additional distillation blocks can reduce the time per T gate down to $1\ominus$. For $p = 10^{-4}$, 11 distillation blocks produce 1 magic state every $1\ominus$. To consume these magic states fast enough, we need to use a fast data block. This fast block uses 231 tiles and the 11 distillation blocks together with their storage tiles use $11 \cdot 12 = 132$ tiles, as shown in Fig. 23a. With a total of 363 tiles, this setup uses 123,000 qubits and finishes the computation

in $10^8\ominus$, i.e., in 21 minutes and 40 seconds.

For $p = 10^{-3}$, parallelizing 5 distillation blocks produces a magic state every $0.936\ominus$. This is faster than the fast block can consume the states, but allows for the execution of a T gate every $1\ominus$. With 231 tiles for the fast block, 405 distillation tiles and 60 storage tiles, the total space cost is 696 tiles. The setup shown in Fig. 20b contains four unused tiles to make sure that all storage lines are connected to the data block. Storage lines need to be connected to the ancilla space of the data block either directly, via other storage lines or via unused tiles. In any case, this corresponds to roughly 1,020,000 physical qubits. The computation finishes after 45 minutes.

Avoiding the classical overhead. Every consumption of a magic state corresponds to a Pauli product measurement, the outcome of which determines whether a Clifford correction is required. This correction is commuted past the subsequent rotations, potentially changing the axis of rotation. Therefore, the computation cannot continue before the measurement outcome is determined. This involves a small classical computation to process the physical measurements (i.e., decoding and feed-forward), which could slow down the quantum computation. In order to avoid this, the magic state consumption can be performed using the auto-corrected $\pi/8$ rotations of Fig. 17b. Here, the classical computation merely determines, whether the ancilla qubit – which we refer to as the correction qubit $|c\rangle$ – is measured in the X or Z basis. While this classical computation is running, the magic state for the subsequent $\pi/8$ rotation can be consumed, as the auto-corrected rotation involves no Clifford correction. This means that distillation blocks should output $|m\rangle - |c\rangle$ pairs, for which we construct modified distillation blocks in the following section. If the classical computation is, on average, faster than $1\ominus$ (i.e., d code cycles), then classical processing does not slow down the quantum computation in the T -count-limited schemes.

Summary. Data blocks combined with distillation blocks can be used for large-scale quantum computing. The first step is to determine a sufficiently high-fidelity distillation protocol. Next, one constructs a minimal setup from a compact data block and a single distillation block to upper-bound the required code distance. Finally, one can trade off space against time by using fast data blocks and adding more distillation blocks. This can reduce the time per T gate down to $1\ominus$. In our example, the trade-off also reduces the space-time cost compared to the minimal setup by a factor of 5 for $p = 10^{-4}$ and by a factor of 2.8 for $p = 10^{-3}$. In order to fully exploit the space-time trade-offs discussed in this section, the input circuit should be optimized for T count.

5 Trade-offs limited by T depth

In the previous section, we parallelized distillation blocks to finish computations in a time proportional to the T count. In this section, we combine the previous constructions of data and distillation blocks to what we refer to as *units*. By parallelizing units, we exploit the fact that, in our example, the 10^8 T gates are arranged in 10^6 layers of 100 T gates to finish the computation in a time proportional to the T depth. We first slightly increase the space-time cost compared to the previous section, in order to speed up the computation down to one measurement per T layer. In this sense, we implement Fowler’s time-optimal scheme [21].

5.1 T layer parallelization

The main concept used to parallelize T layers is quantum teleportation. The teleportation circuit is shown in Fig. 24a. It starts with the generation of a Bell pair $(|00\rangle + |11\rangle)/\sqrt{2}$ by the $Z \otimes Z$ measurement of $|+\rangle \otimes |+\rangle$. An arbitrary gate U is performed on the second half of the Bell pair. Next, a qubit $|\psi\rangle$ and the first half of the Bell pair are measured in the Bell basis, i.e., in $X \otimes X$ and $Z \otimes Z$. After the measurement, the first two qubits are discarded and $|\psi\rangle$ is teleported to the third qubit through the gate U . This means that the output state is $U|\psi\rangle$, if the teleportation is successful. However, it is only successful, if both Bell basis measurements yield a +1 outcome. In the other three cases, the teleported state is $UX|\psi\rangle$, $UY|\psi\rangle$ or $UZ|\psi\rangle$. Note that the correction operation to recover the state $|\psi\rangle$ is not a Pauli operation P , but instead UPU^\dagger , which, in general, is as difficult to perform as U itself.

If U is a $P_{\pi/8}$ rotation, as in Fig. 24b, the Pauli errors change $P_{\pi/8}$ to $P_{-\pi/8}$ up to a Pauli correction. Since it is only after the Bell basis measurement that

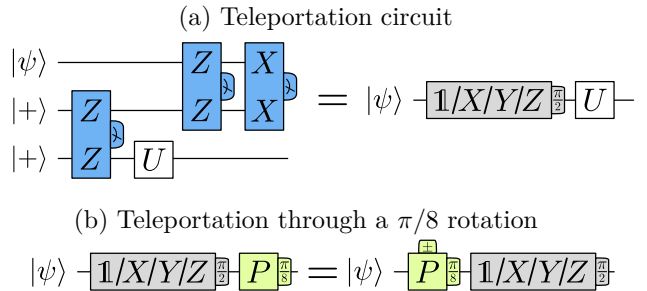


Figure 24: (a) Circuit for quantum teleportation of $|\psi\rangle$ through a gate U . Only if both Bell basis measurements yield +1, the teleported state is $U|\psi\rangle$. If $Z \otimes Z = -1$, the state is $UX|\psi\rangle$. If $X \otimes X = -1$, the state is $UZ|\psi\rangle$. If both measurements yield -1, the state is $UY|\psi\rangle$. (b) If U is a $\pi/8$ rotation, the corrective Paulis change $P_{\pi/8}$ to $P_{-\pi/8}$.

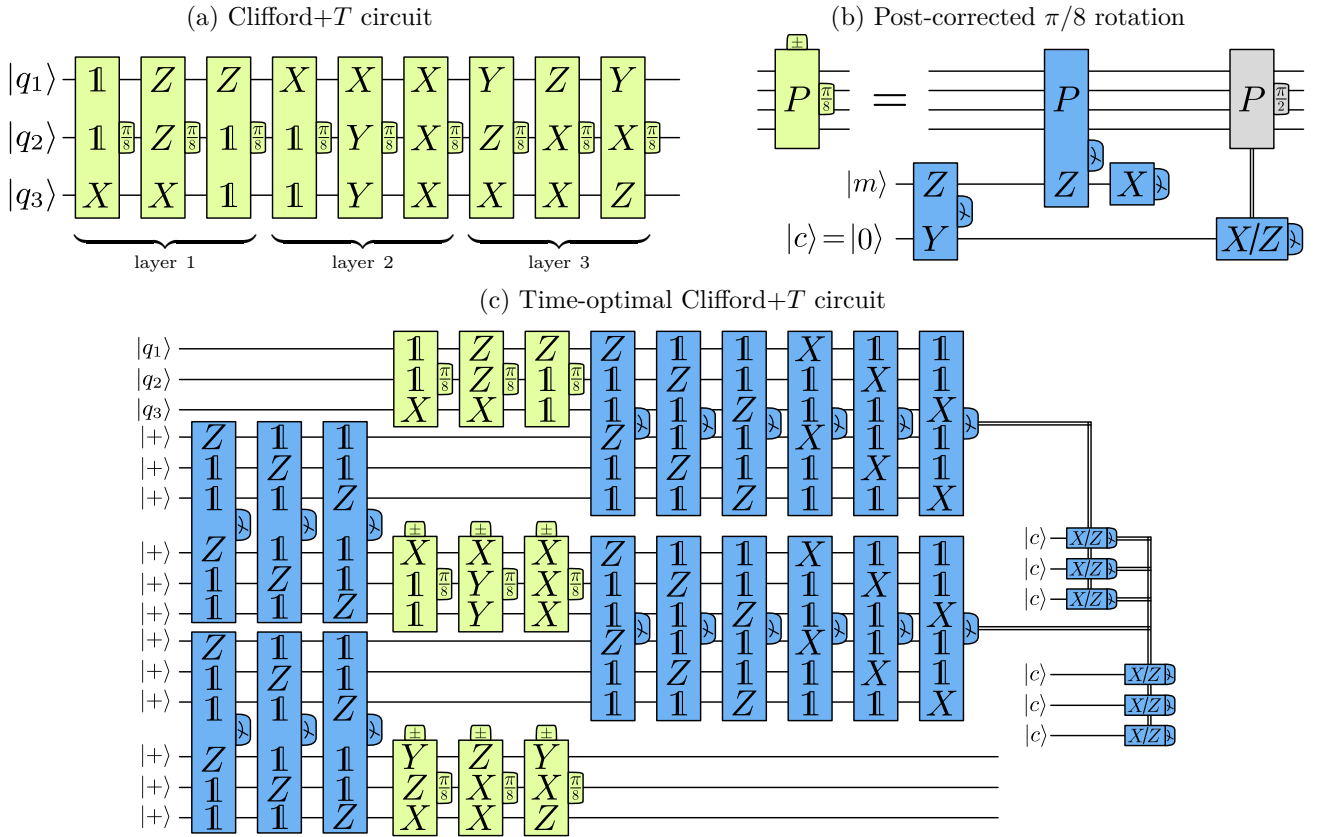


Figure 25: Time-optimal implementation of a three-qubit quantum computation consisting of 9 T gates in 3 T layers. Post-corrected $\pi/8$ rotations (b) can be used to decide at a later point, whether the performed operation was a $P_{\pi/8}$ or a $P_{-\pi/8}$ rotation.

we know, whether we should have performed a $P_{\pi/8}$ or a $P_{-\pi/8}$ gate, we use post-corrected $\pi/8$ rotations in Fig. 25b, which are similar to the auto-corrected rotations of Fig. 17b. The post-corrected rotation uses a resource state consisting of two qubits, a magic state $|m\rangle$ and a second qubit that we refer to as a correction qubit $|c\rangle$. The resource state is generated by initializing $|c\rangle$ in $|0\rangle$ and measuring $Z \otimes Y$ between $|m\rangle$ and $|c\rangle$. In order to perform a post-corrected $\pi/8$ rotation, the resource state is consumed by measuring $P \otimes Z$ involving the magic state, and measuring $|m\rangle$ in X . The correction qubit $|c\rangle$ is stored for later use. It can be used at a later moment to decide, whether the rotation should have been a $+\pi/8$ or $-\pi/8$ rotation by measuring $|c\rangle$ either in the Z or X basis. Depending on the measurement outcome, a Pauli correction may be required.

The time-optimal circuit. This can be used to execute multiple T layers simultaneously. If U is a product of mutually commuting $\pi/8$ rotations, i.e., a T layer, the teleportation corrections replace all $\pi/8$ rotations with post-corrected rotations. An example is shown in Fig. 25 for a three-qubit computation of three T layers,

where all three T layers are executed simultaneously. The reason why we can only group up T gates that are part of the same layer is that otherwise the Pauli corrections of the post-corrected rotation would not commute with the other rotations. The time-optimal circuit consists of three steps: The preparation of Bell pairs for each T layer, the application of T gates, and a set of final Bell measurements. At this point, the computation is not finished, as we still need to measure the correction qubits of the post-corrected rotations. Because these involve potential Pauli corrections, the correction qubits of the different T layers need to be measured one after the other. Thus, every T layer is executed one after the other, where each execution requires the time that it takes to measure the correction qubits and perform the classical processing to determine the next set of measurements from the Pauli corrections. We refer to this time as t_m . In other words, any Clifford+ T circuit consisting of n_L T layers can be executed in $n_L \cdot t_m$, independent of the code distance, which is the main feature of the time-optimal scheme [21].

The circuit in Fig. 25c naively requires $2n \cdot n_L$ qubits

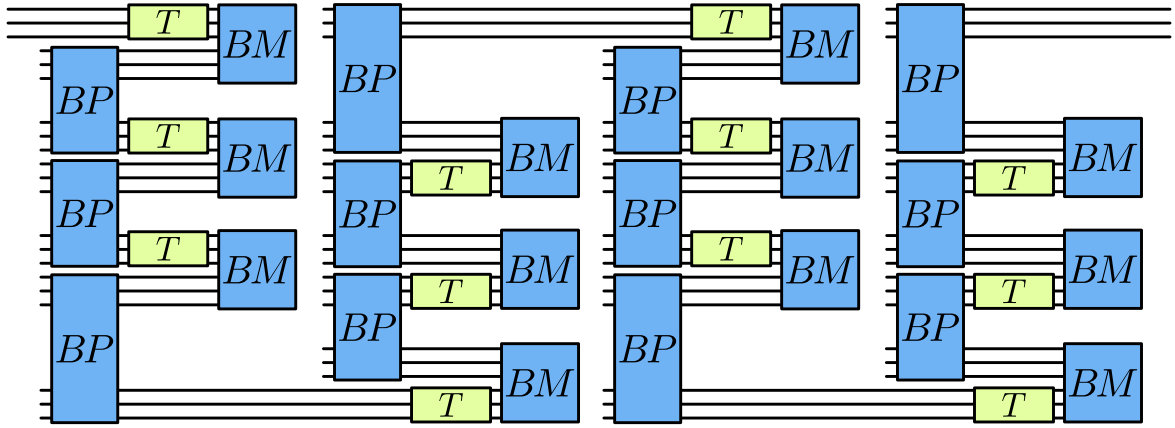


Figure 26: An example of a time-optimal circuit using four units. In this case, each unit consists of six qubits, i.e., it is a three-qubit quantum computation, where three T layers can be executed simultaneously.

for an n -qubit computation, which scales with the length of the computation. Since we only have a finite number of qubits at our disposal, our goal is to implement the circuit in Fig. 26 instead. Here, the qubits form groups of $2n$ qubits. We refer to each of these groups as a *unit*. Using n_u units, $n_u - 1$ layers of T gates can be performed at the same time. In the circuit, the steps of Bell state preparation (BP), post-corrected T layer execution (T) and Bell basis measurement (BM) are performed repeatedly until the end of the computation. We refer to the block of operations (BP - T - BM) as *unit preparation*. Every time that unit preparation is finished, all qubits except for the correction qubits (not shown in Fig. 26) and half of the qubits of the last unit are discarded. At this point, the next set of unit preparations begins. Simultaneously, the correction qubits of the recently finished units are measured one after the other, which has a time cost of $(n_u - 1) \cdot t_m$. This means that the number of units can be increased to speed up the computation, until $(n_u - 1) \cdot t_m$ reaches the time that it takes to prepare a unit t_u . At this maximum number of units $n_{\max} = t_u/t_m + 1$, a T layer is executed every t_m and the computation cannot be sped up any further in the Clifford+ T framework.

Note that the first and last unit differ from the other units. While all other units need to execute n_T T gates every t_u , the first and last unit need to execute n_T T gates only every $2t_u$, where n_T is the number of T gates per layer. Furthermore, the other blocks need to be able to store up to $2n_T$ correction qubits, since, after the end of a unit preparation, n_T correction qubits are stored, and may need to remain stored until the end of the next unit preparation. For the first and last block, on the other hand, the required storage space is halved.

In the following, we will show how to prepare units in our framework. We find that, for our examples, unit

preparation takes $113\odot$. If $t_m = 1 \mu\text{s}$, then n_{\max} is ~ 1500 for $p = 10^{-4}$ and ~ 3000 for $p = 10^{-3}$. Independently of the error rate, the computational time drops to one second.

5.2 Units

Units differ from the fast setups in Fig. 23 in three aspects. First, the number of qubits stored in the data block is doubled. Secondly, the distillation protocols are modified to output $|m\rangle$ - $|c\rangle$ pairs, instead of just magic states $|m\rangle$. Thirdly, in order to store correction qubits $|c\rangle$, additional space is required. Contrary to magic-state storage tiles, correction-qubit storage tiles do not need to be connected to the data block's ancilla region.

Modified distillation blocks. In order to have distillation blocks output $|m\rangle$ - $|c\rangle$ pairs, extra tiles and operations are required. We show the necessary modifications for the example of 15-to-1 and 116-to-12 distillation. A modified 15-to-1 block is shown in Fig. 27a. Apart from the standard 11 distillation tiles (orange) and one magic-state storage tile (green), it also contains 19 correction-qubit storage tiles (purple) and an additional tile (gray) that is used for neither distillation nor storage. The additional steps that modify the protocol are shown in Fig. 27c, which zooms into the highlighted region of Fig. 27a. In step 1 of the shown protocol, the distillation has just finished after $11\odot$. The patch of the output state is deformed in step 2, and an additional qubit $|c\rangle$ is initialized in the $|0\rangle$ state. The $Y \otimes Z$ operator between $|c\rangle$ and $|m\rangle$ is measured in step 3. In step 4, the correction qubit is sent to storage. Finally, in step 5, the magic state $|m\rangle$ is moved to its storage tile. This operation blocks one of the orange tiles that is used for the distillation protocol for $4\odot$. Still, this does not slow down 15-to-1 distillation, since the first 4 rota-

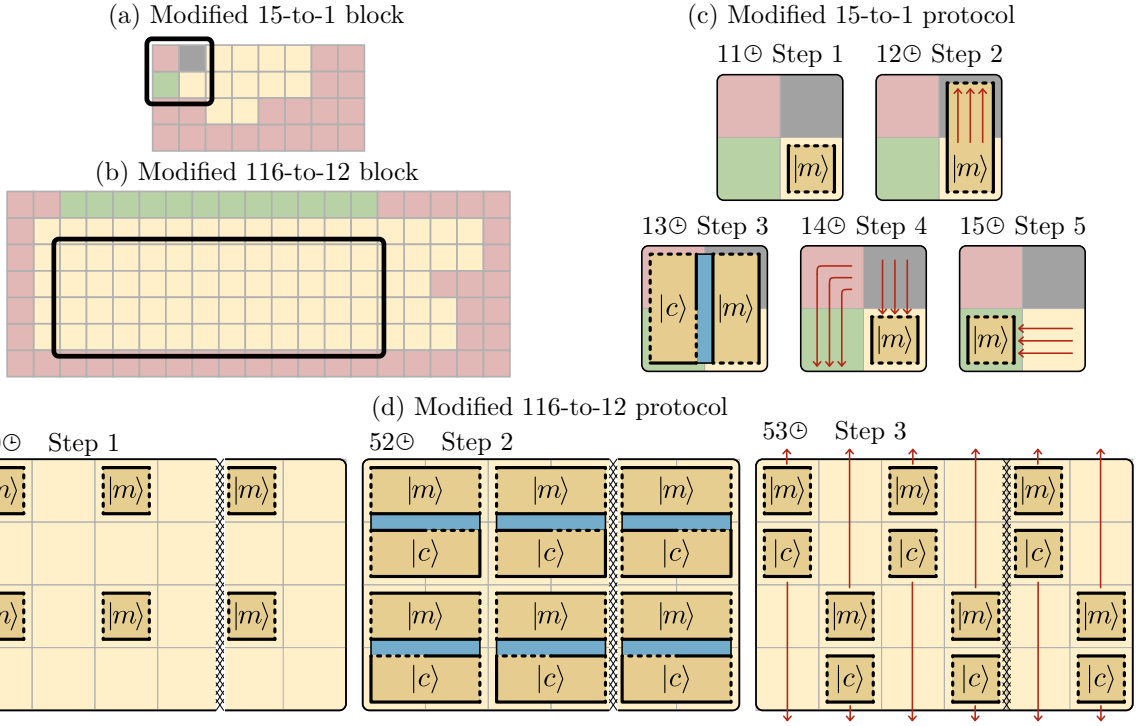


Figure 27: Modified 15-to-1 distillation blocks (a) output a $|m\rangle$ - $|c\rangle$ pair every $11\ominus$. After the end of the distillation protocol, four additional steps (c) are necessary. The modified 116-to-12 distillation block (b) finishes after $53\ominus$, due to the three additional steps in (d).

tion of the protocol in Fig. 15 can be chosen, such that the output qubit is not needed. Therefore, the modified distillation block outputs one $|m\rangle$ - $|c\rangle$ pair every $11\ominus$.

For 116-to-12 distillation, a modified block is shown in Fig. 27b. We arrange the qubits, such that the 12 output states are found in the positions shown in step 1 of Fig. 27d. Using $2\ominus$, correction qubits are prepared and $Y \otimes Z$ operators are measured. Finally, the patches are deformed back to square patches and all magic states are sent to the green storage, while all correction qubits are sent to the purple storage. This adds $3\ominus$ to the protocol, meaning that this block outputs 12 $|m\rangle$ - $|c\rangle$ pairs every $53\ominus$ with a success probability of $(1-p)^{116}$. For $p = 10^{-3}$, this corresponds to one output every $4.96\ominus$.

As mentioned in Sec. 4, modified distillation blocks can also be used with setups, in which T gates are performed one after the other, in order to deal with slow classical processing. In this case, only one correction qubit storage tile per magic state is required.

Units. Modified distillation blocks together with fast data blocks are what we refer to as units. The units for our example computation for $p = 10^{-3}$ and $p = 10^{-4}$ are shown in Fig. 29a-b. They both consist of a 200-qubit fast data block, 200 correction-qubit storage tiles, and a number of distillation blocks. Since we will show that unit preparation takes $113\ominus$ in our case, the num-

ber of distillation blocks is chosen such that at least 100 $|m\rangle$ - $|c\rangle$ pairs can be distilled in $113\ominus$. A full time-optimal quantum computer consists of a row of multiple units, see Fig. 29c. The units shown in the figure contain some unused tiles. This gives the units a rectangular profiles, even though this is not necessarily required. In our case, the units have a footprint of 54×21 and 37×21 tiles, respectively. Note that the first and last

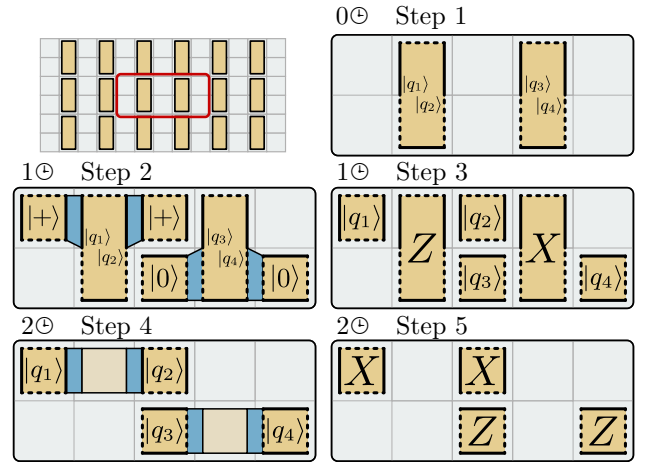


Figure 28: Bell basis measurement (BM) in $2\ominus$.

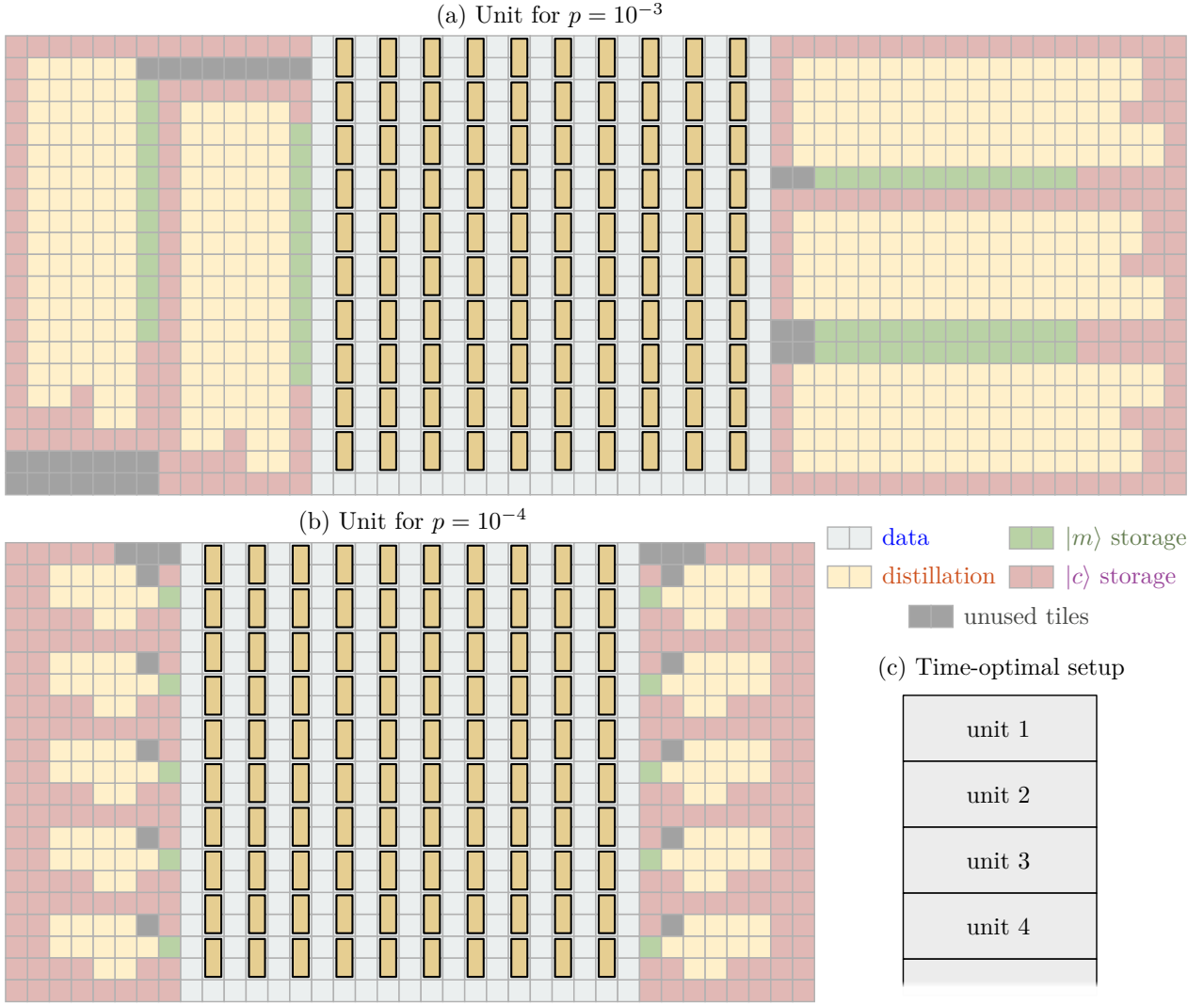


Figure 29: Units consist of fast data blocks, modified distillation blocks and storage tiles. (a) The unit for $p = 10^{-3}$ consists of $54 \times 21 = 1134$ tiles. (b) For $p = 10^{-4}$, the number of tiles is $37 \times 21 = 777$. (c) A time-optimal setup consists of a row of multiple units, which means that the space to the bottom and top of the fast data blocks needs to remain free.

unit of a time-optimal setup are smaller, as they only require 100 correction-qubit storage tiles and half the number of distillation blocks.

Unit preparation. In order to implement the time-optimal circuit of Fig. 26 with the setup of Fig. 29, we show protocols that can be used for the *BP-T-BM* operations. The data blocks of every unit store $2n$ qubits in n two-qubit patches. We arrange the qubits in such a way that the the final Bell measurements (*BM*) are $Z \otimes Z$ and $X \otimes X$ measurements of the two qubits of every two-qubit patch. This Bell measurement can be done in $2\ominus$, as shown in Fig. 28.

This arrangement of qubits implies that, for every two-qubit patch, one of the qubits needs to be part of a Bell state preparation (*BP*) with the neighboring unit

to the top, and the other with a neighboring unit to the bottom. For an n -qubit quantum computation, this Bell state preparation can be performed in $\sqrt{n}+1$ time steps, as we show in Fig. 30 for the example of $n = 9$. For this, every qubit is initialized in the $|+\rangle$ state. The Bell state preparation requires a series of $Z \otimes Z$ measurements. The protocol in Fig. 30 shows that, since an n -qubit computation implies that the number of rows of the data block is \sqrt{n} , these measurements require a total of $\sqrt{n} + 1$ time steps.

In total, the unit preparation of an n -qubit computation with n_T T gates per layer requires $\sqrt{n}+1$ time steps for the Bell state preparation, n_T time steps for the execution of the T layer, and 2 time steps for the Bell basis measurement, i.e., a total of $n_T + \sqrt{n} + 3$ time steps. In

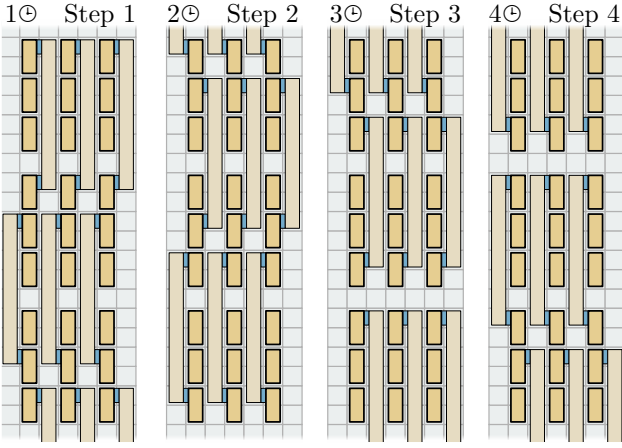


Figure 30: Bell state preparation (BP) for a 9-qubit computation (18 qubits per unit) in $4\ominus$. All two-qubit patches are initialized in the $|+\rangle^{\otimes 2}$ state. Each measurement ancilla is used for a $Z \otimes Z$ measurement between two qubits in different units. For n -qubit computations, this requires $\sqrt{n} + 1$ time steps.

our example, this amounts to $113\ominus$, which corresponds to $t_u = 1469 \mu\text{s}$ for $p = 10^{-4}$ and $t_u = 3051 \mu\text{s}$ for $p = 10^{-3}$. Thus, time optimality is reached with 1470 units for $p = 10^{-4}$ and 3052 units for $p = 10^{-3}$.

Space-time trade-offs. Of course, it is also possible to use fewer units than required for time optimality. Using n_u units means that $n_T \cdot (n_u - 1)$ T gates are performed every t_u . In our example, $100 \cdot (n_u - 1)$ T gates are performed every $113\ominus$. With three units, the computational time drops to 56.5% of the computational time of the fast setup in Fig. 23. With ten units, it drops to 11%. The number of qubits per unit is $\sim 260,000$ for $p = 10^{-4}$ and $\sim 1,650,000$ for $p = 10^{-3}$, so going from the fast setup to parallelized units is, initially, not a favorable space-time trade-off. Since the space-time cost has increased compared to the fast setup, it is also useful to check whether the code distance needs to be readjusted. If we use three units – ignoring that the first and last unit are, in principle, smaller – the space-time cost is still below the space-time cost of the minimal setup in both cases. Adding more units significantly improves the space-time cost. It is also a prescription to linearly speed up the quantum computer down to the time-optimal limit.

5.3 Distributed quantum computing

Note that, apart from the initial sharing of entangled Bell pairs, the units operate entirely independently of each other. This implies that, if Bell pairs can be shared between different quantum computers, each unit can be located in a separate quantum computer. The shared Bell pairs do not even need to have a high fidelity, as

(a) Distributed quantum computing

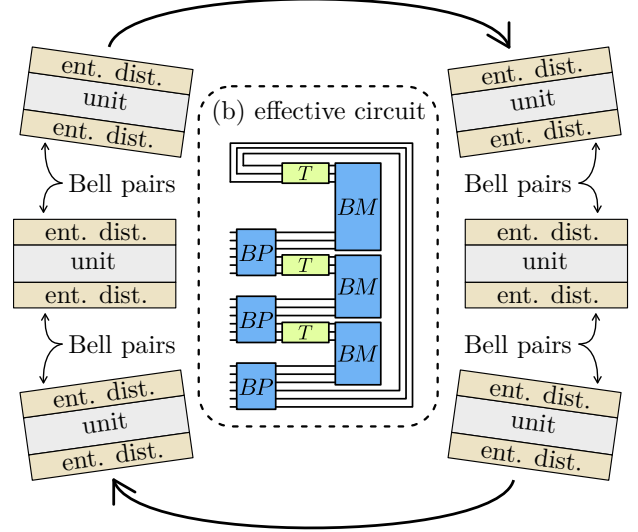


Figure 31: Scheme for distributed quantum computing in a circular arrangement of quantum computers with the ability to share Bell pairs between nearest neighbors. If the Bell-pair fidelity is low, entanglement distillation (ent. dist.) can be used to increase the fidelity. This scheme effectively implements the circular time-optimal circuit drawn schematically in (b).

software-based entanglement distillation [39, 40] can be used to convert a large number of low-fidelity Bell pairs into fewer high-fidelity Bell pairs. Recent experiments have made progress towards generating entanglement between different superconducting chips [41–43].

For the time-optimal scheme, quantum computers may be arranged in a circle as shown in Fig. 31a, with the ability to share Bell pairs between neighboring quantum computers. This effectively implements the circuit that is schematically drawn in Fig. 31b. Note that in this circuit, there is no first and last unit. Here, every unit performs $n_T \pi/8$ rotations every t_u . Therefore, time optimality is reached with one fewer unit, and each unit only needs to store n_T correction qubits instead of $2n_T$. With only 100 correction-qubit storage tiles and ignoring the unused tiles, the qubit count of the units in Fig. 29 drops to $\sim 220,000$ for $p = 10^{-4}$ and $\sim 1,470,000$ for $p = 10^{-3}$, which are the numbers that we report in Fig. 3. Thus, if nearest-neighbor communication between quantum computers is feasible, already fewer than 2 million physical qubits per quantum computer can be used to implement the full time-optimal scheme with 1500-3000 quantum computers.

Entanglement distillation increases the qubit count. Note that it does not slow down the computation, as Bell pairs do not need to be distilled instantly. Entanglement distillation can take up to t_u to distill the n_T Bell pairs required per entanglement distillation block.

Summary. In order to speed up an n -qubit quantum computation beyond 1Θ per T gate, we parallelize T layers using units. With an average of n_T T gates per layer, a unit consist of $4n + 4\sqrt{n} + 1$ tiles for the data block, $2n_T$ storage tiles for the correction qubits, and enough distillation blocks to distill n_T $|m\rangle$ - $|c\rangle$ pairs in the time it takes to prepare a unit, which is $n_T + \sqrt{n} + 3$ time steps. If the unit preparation time is t_u and the time for single-qubit measurements and classical processing is t_m , a time-optimal setup consists of $t_u/t_m + 1$ units, executing one T layer every t_m . Using fewer units results in a linear space-time trade-off. With n_u units, $n_T \cdot (n_u - 1)$ T gates are performed in t_u . A circular arrangement of units can be used for distributed quantum computing. This also reduces the number of correction-qubit storage tiles to $1n_T$ and the number of units in a time-optimal setup to t_u/t_m . In order to fully exploit the space-time trade-offs discussed in this section, the input circuit should be optimized for T depth.

6 Trade-offs beyond Clifford+ T

Under the assumption that measurements and feed-forward can be done in $1 \mu\text{s}$, we described how to perform a 10^8 - T -gate computation in just 1 second. A more conservative assumption would be a measurement and feed-forward time of $10 \mu\text{s}$, which increases the computation time to 10 seconds. Although this seems fast, many quantum computations have T counts that are significantly higher than 10^8 . While the T count of Hubbard model simulations [2] is indeed in this range, quantum chemistry simulations can be more demanding. In particular, the simulation of FeMoco [1], a structure that plays an important role in nitrogen fixation, can have a T count of up to 10^{15} . With a serial execution of one T gate every $10 \mu\text{s}$, the computation takes 317 years to finish. Even if the gates are grouped into 100 T gates per layer, the computation still takes over 3 years.

While Clifford+ T is a gate set that is very well suited for surface codes, it is often not the gate set which is natural to the quantum computations in question. In particular, quantum simulation based on Trotterization consists of many small-angle rotations. In the Clifford+ T framework, each small-angle rotation is translated into a series of T gates via gate synthesis. Depending on the desired precision, this can require ~ 100 T gates for each rotation [44], which must be executed in series. In order to speed up computations beyond their T count or T depth, it is therefore constructive to consider additional resources for gates other than T gates.

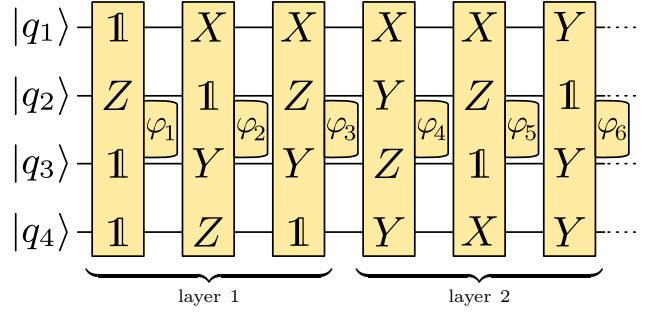


Figure 32: Clifford+ φ circuit. The first two rotation layers (φ layers) with three rotations per layer are shown.

6.1 Clifford+ φ circuits

Instead of requiring an input circuit that consists of Clifford gates and $\pi/8$ rotations, we consider circuits that consist of Clifford gates and arbitrary φ rotations, which we call Clifford+ φ circuits. Using the procedure in Sec. 1, Clifford gates can be commuted to the end of the circuit, such that we end up with a circuit like the one in Fig. 32. Rotations that mutually commute can be grouped up into layers. The algorithm of Sec. 1 can be used to reduce the number of layers. It can even reduce the number of rotations, since, if two rotations P_{φ_1} and P_{φ_2} with the same axis of rotation are moved into the same layer, they can be combined into a single rotation $P_{\varphi_1+\varphi_2}$. Clifford+ φ circuits are characterized by their *rotation count* (or φ count) and *rotation depth* (or φ depth), rather than T count and T depth.

Each φ rotation can be performed using a $|\varphi\rangle = |0\rangle + e^{i(2\varphi)}|1\rangle$ resource state. When this state is consumed to perform a P_φ rotation, there is a 50% chance that a $P_{-\varphi}$ rotation is performed instead. For $\pi/8$ rotations, this is not very problematic, since the correction operation is a $\pi/4$ rotation, which can simply be commuted to the end of the circuit. For general $P_{-\varphi}$, the correction is a $P_{2\varphi}$ rotation, which requires the use of a $|2\varphi\rangle$ state. If this fails, the next correction is a $P_{4\varphi}$ rotation requiring a $|4\varphi\rangle$ state and so on. Thus, a wide variety of resource state is required to execute arbitrary-angle rotations. In the case of $\varphi = \pi/2^k$ for an integer k , $|\varphi\rangle$ states can be distilled using specialized protocols [35, 45]. For other angles, $|\varphi\rangle$ states can be approximated using $|\pi/2^k\rangle$ states, or pieced together from ordinary magic states $|m\rangle$ via circuit synthesis. Ordinary magic states can also generate states that can be used for V gates [46–48], which are Pauli rotations with an angle $\theta = \arccos(3/5)$.

All the schemes discussed in this work can be used with Clifford+ φ circuits by replacing magic state distillation blocks by distillation blocks that produce resource states for arbitrary-angle rotations. In order to consume these states in a systematic way similar to the

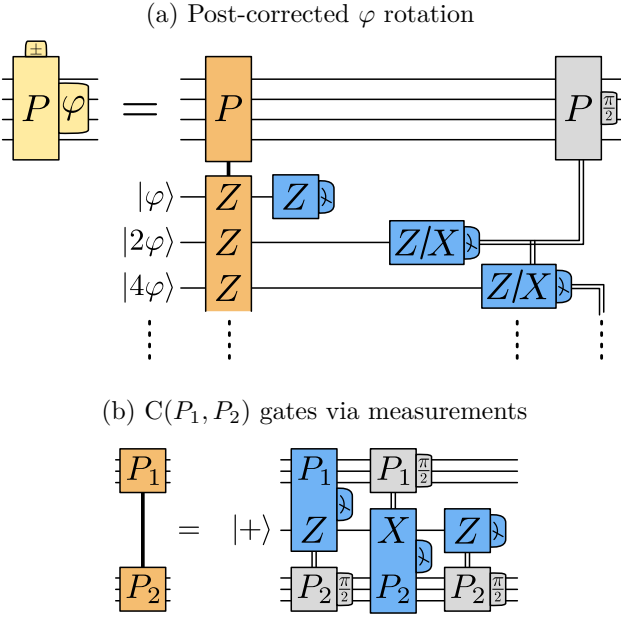


Figure 33: (a) A post-corrected φ rotation can be used to decide at a later point, whether the performed operation was a P_φ or a $P_{-\varphi}$ gate. (b) A $C(P_1, P_2)$ gate can be performed explicitly using a $|+\rangle$ ancilla and Pauli product measurements.

post-corrected $\pi/8$ rotations in Fig. 25b, we can use the post-corrected version of φ rotations shown in Fig. 33. First, the n resource states are entangled with the data qubits via a $C(P, Z^{\otimes n})$ gate. Just like magic state consumption, this can be done every $1\ominus$, since the data qubits are only part of one measurement in the measurement circuit in Fig. 33b. Next, the $|\varphi\rangle$ state is measured in Z . If the outcome of this measurement is $+1$, then the rotation is successful and all other resource states are discarded by measuring them in X . If, instead, the outcome is -1 , the $|2\varphi\rangle$ state is measured in Z . If the outcome of this Z measurement is $+1$, the correction is successful, and the remaining resource states are discarded by X measurements. For -1 , the corrections continue with a Z measurement of $|4\varphi\rangle$. Note that, in most cases, this cascade of measurements finishes in the second step. Therefore, on average, it takes $2t_m$ to perform these measurements. However, sufficiently many resource state are required in order to be prepared for the most unlikely situations, in which many measurement steps are required. The probability to require n measurement steps (i.e., n resource states down to $|2^n\varphi\rangle$) is exponentially low, 2^{-n} . Therefore, the number of resource states that need to be generated for each φ rotation scales logarithmically with the rotation count of the circuit, if one wants to stay below a certain probability that any of these rotations is slowed down by a missing resource state. If

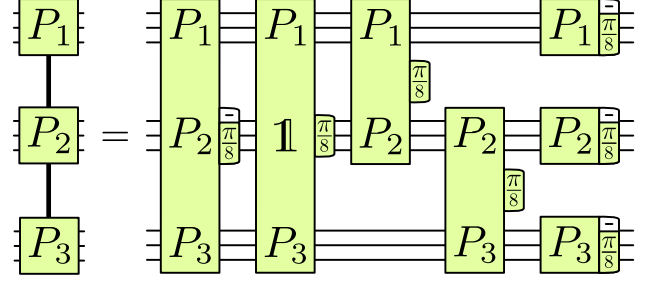


Figure 34: $C(P_1, P_2, P_3)$ gate in terms of seven $\pi/8$ rotations.

$|\pi/2^k\rangle$ states are used, the cascade of measurements terminates after k steps. This technique of cascading resource state measurements is also referred to as programmable ancilla rotations [49]. Note that the cascade of measurements can also be postponed to a later point, such that the post-corrected φ rotations can be used in the time-optimal scheme.

Using the T -count-limited scheme of Sec. 4, we can execute a φ rotation every $1\ominus$. For 100 T gates per φ rotation, this speeds up the computation by a factor of 100. Also, the time-optimal setting of Sec. 5 can be used with Clifford+ φ circuits. However, the execution of a φ layer can take more than $2t_m$, as the measurement cascades for all rotations in the layer need to terminate. For instance, for 100 rotations per layer, each layer execution takes, on average, $8t_m$. For 100 T gates per rotation, φ layer parallelization reduces the computational time by a factor of 12.5 compared to T layer parallelization, i.e., from over 3 years to 3 months. In the specific case of quantum chemistry simulations, their T count can be reduced significantly by using more advanced algorithms [50–52], which also profit from arbitrary-angle rotations. Thus, if distributed quantum computing is feasible, Clifford+ φ circuits such as the ones used for quantum chemistry can be executed with qubit counts per quantum computer not far above the numbers reported in Fig. 3. The only difference to Clifford+ T units is that larger distillation blocks are required to produce and store the $|\varphi\rangle$ resource states.

Multi-controlled Pauli gates. Other gates that are used extensively in quantum algorithms are multi-controlled Paulis, such as Toffoli or CCZ gates. In Fig. 5, we have shown how $C(P_1, P_2)$ gates can be written in terms of $\pi/4$ rotations. A similar decomposition is possible for multi-controlled Pauli gates. In Fig. 34, we show how a $C(P_1, P_2, P_3)$ gate is a product of 7 $\pi/8$ rotations. For instance, $C(Z, Z, X)$ is the Toffoli gate. From the circuit, it is evident that the T depth of $C(P_1, P_2, P_3)$ gates is one [28]. In principle, these doubly-controlled Pauli gates can be written with just four T gates [53], but this increases the number of layers and a similar effect can be obtained by cancelling

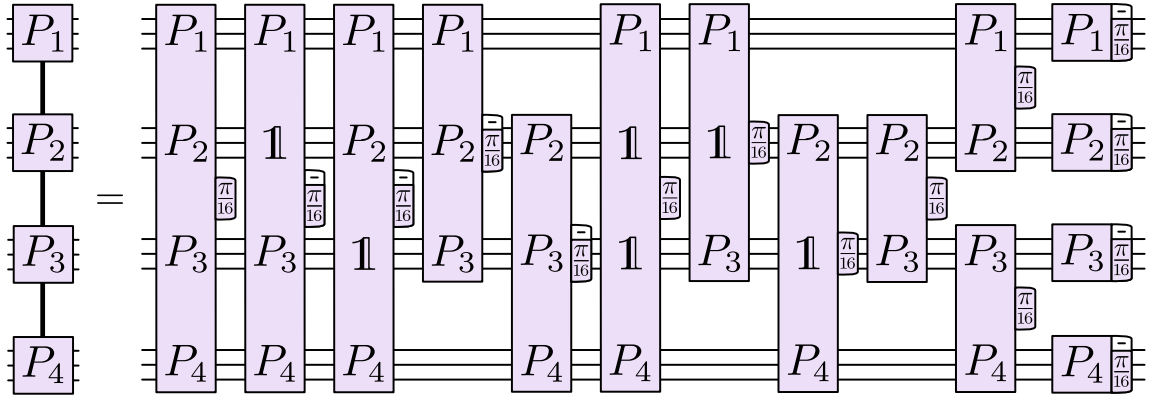


Figure 35: $C(P_1, P_2, P_3, P_4)$ gate in terms of 15 $\pi/16$ rotations.

$\pi/8$ rotations from pairs of doubly-controlled gates in a circuit. Reducing the T count by increasing the circuit depth [54] can still be a useful circuit manipulation for T -count-limited setups. We also note that the T count can be reduced by combining gate synthesis and magic state distillation (*synthillation*) [55, 56].

$C(P_1, P_2, P_3, P_4)$ gates, i.e., triply-controlled Pauli gates, can be written as 15 $\pi/16$ rotations, as shown in Fig. 35. While the T depth of this circuit is no longer 1, the *rotation* depth is. In fact, any multi-controlled Pauli gate with n controls can be constructed from $2^n - 1$ $P_{\pi/2^n}$ rotations by following the pattern shown in Figs. 5, 34 and 35. The rotation depth of all these gates is 1. Multi-controlled gates can also be pieced together from $C(P_1, P_2, P_3)$ rotations, but this increases the circuit depth. By using small-angle rotations, any multi-controlled Pauli gate can be executed in one step.

6.2 Shorter measurements

If the bottleneck of slow classical processing can be overcome, then the only hardware-based restriction to the speed of quantum computation is the time it takes to measure a physical qubit. In the time-optimal scheme, the execution time of each rotation layer is governed by the measurement time. This measurement time only needs to be high, if the measurement fidelity is required to be sufficiently low. In order to speed up the computation, one can use shorter qubit measurements. This exponentially decreases the measurement fidelity. On the other hand, the measurement fidelity of encoded surface-code qubits increases exponentially with the number of qubits comprising the logical qubit. Thus, by using twice as many physical qubits to encode the measured logical qubit, the measurement time can be decreased by a factor of two, doubling the computational speed of the quantum computer. In fact, not

all qubits need to use a higher code distance. Only the correction qubits that are measured to execute each rotation layer need to be larger, and only right before they are measured. The physical qubit measurement does not need to be a quantum non-demolition measurement, but can be a destructive measurement. Ultimately, however, the speed of quantum computation is limited by the speed of classical computation. Exploring superconducting logic [57] to speed up classical computation may be a viable route to speed up quantum computers.

Summary. All the schemes discussed in this paper can not only be used with Clifford+ T circuits, but also with Clifford+ φ circuits. The only difference is that more and different resource states are required. Their distillation and storage requires more space than ordinary magic state distillation, but their use can speed up the computation by several orders of magnitude.

7 Conclusion

In this work, we described how full quantum computations can be performed in surface-code-based architectures of different sizes. Previous works on the translation of quantum computations into surface-code schemes [36, 58–60] attempted to optimize the logical qubit arrangement via algorithms that take a quantum circuit as an input. Here, we took a different approach by discussing computational schemes that do not require any prior knowledge about the input circuit. This has the advantage that a resource count with our schemes only requires the T count and T depth of the input circuit, and that the schemes consist of modular blocks that can be optimized independently of each other. In addition, the space-time cost is lower compared to earlier works [20, 36].

Big quantum computers are fast. Starting from the minimal setup in Fig. 21 that consists of a compact

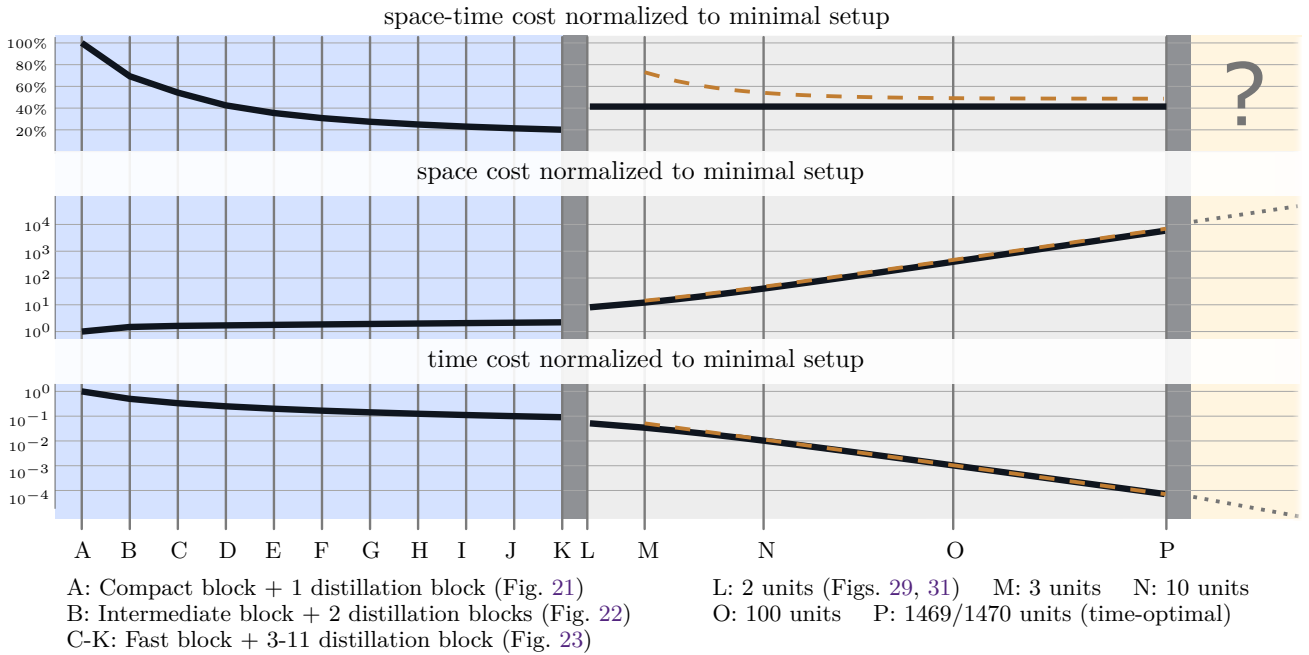


Figure 36: Space-time, space, and time cost of the schemes discussed in this paper for the example of a 100-qubit quantum computation with T count 10^8 and T depth 10^6 , under the assumption of a $1 \mu\text{s}$ code cycle time, and a $1 \mu\text{s}$ measurement and classical processing time. The solid and dashed lines in M-P are for circular (solid) and linear (dashed) arrangements of units.

data block and a single distillation block, we traded off space versus time, increasing the size of the quantum computer and, in return, decreasing the computational time. For the example of a computation with a T count of 10^8 and a T depth of 10^6 with an error rate of $p = 10^{-4}$, the minimal setup consists of 164 tiles and executes one T gate every $11\ominus$, corresponding to a computational time of 4 hours with 55,400 physical qubits. From here, the space-time cost is drastically reduced by adding more distillation blocks, as shown in Fig. 36 and Tab. 2. With this strategy, the computational time is reduced to $1\ominus$ per T gate, where the computational cost of a circuit is governed by its T count.

For further space-time trade-offs, we parallelized T layers using units. This is an increase in space-time cost, especially for linear arrangements of units (dashed line in Fig. 36), but enables further space-time trade-offs. Linearly trading off space versus time, the computational time can be reduced to one measurement per T layer. Units are well-suited for distributed quantum computing, as the sharing of Bell pairs between neighboring units is part of the parallelization scheme.

This exhausts the space-time trade-offs that are possible within the Clifford+ T framework. Switching to Clifford+ φ circuits can provide further trade-offs, as additional resources are introduced for arbitrary-angle rotations. This can be used to execute circuits in a time proportional to their rotation depth, as opposed to their

T depth. We have not investigated how this trade-off affects the space-time cost in our scheme.

Room for optimization. In our T -count-limited schemes and for the preparation of units, one T gate is performed after the other. If the input circuit is known, it is reasonable to assume that qubits can be arranged in a way that allows for the parallel execution of multiple T gates in the same data block. Furthermore, there is a strict separation between tiles used for magic state distillation and tiles used for data blocks in our schemes. By sharing tiles between blocks, the space overhead may be reduced. Moreover, we have only considered a handful of distillation protocols. It would be interesting to see which distillation protocols can be used to optimize the cost function of Eq. (9). Finally, concrete tile layouts that can be used to distill and consume the additional resources necessary for Clifford+ φ computing are still missing.

Beyond surface codes. Even though we designed our schemes with surface codes in mind, they can, in principle, be applied to other toric-code-based patches, such as Majorana surface-code patches [11] or color-code patches [13, 61, 62]. Color codes can reduce the number of physical qubits due to more compact encoding, but require more elaborate hardware to measure the higher-weight check operators. The space cost is reduced by replacing all surface-code patches by color-code patches, with the exception of Pauli product mea-

scheme	A	B	C-K	L	M	N - P
physical qubits	55,400	76,400	90,200 - 123,000	447,000	679,000 (788,000)	2,230,000 - 328,000,000 (2,630,000 - 386,000,000)
computational time	4 h	2 h	79-22 min	12 min	490 sec (734 sec)	147 sec - 1 sec (163 sec - 1 sec)

Table 2: Space and time cost of the schemes plotted in Fig. 36. The number in parentheses are for linear arrangements of units (dashed lines in Fig. 36).

surement ancillas. In order to keep the space cost low, measurement ancillas should remain surface-code patches and color-to-surface code lattice surgery [63] should be used during the Pauli product measurement protocol, as described in Ref. [64].

Outlook. If the number of qubits continues to double every 8 months [65], the 60,000 - 300,000 physical qubits necessary for classically intractable Hubbard model simulations with a T count of 10^8 will be available in 7-9 years, assuming qubit quality improves accordingly. If multiple quantum computers can be connected in a network, time-optimal quantum computing becomes available shortly thereafter, facilitating the implementation of more difficult algorithms such as quantum chemistry simulations or Shor’s algorithm. Classical processing in terms of measurements, feed-forward and decoding is expected to be a significant roadblock in speeding up quantum computers. Ultimately, faster classical control hardware will be necessary to build faster quantum computers. I hope that the schemes discussed in this work are a useful roadmap towards large-scale quantum computing, and that the patch-based framework is a valuable toolbox for constructions of surface-code-based implementations of quantum algorithms.

Acknowledgments

This work would not have been possible without insightful discussion with Austin Fowler and Craig Gidney about Pauli product measurements and 15-to-1 distillation, with Jens Eisert, Markus Kesselring and Felix von Oppen about Clifford tracking and space-time trade-offs, with Jeongwan Haah and Matthew Hastings about magic state distillation, with Guang Hao Low and Nathan Wiebe about quantum simulation algorithms, and with Ali Lavasani about few-qubit surface-code architectures. This work has been supported by the Deutsche Forschungsgemeinschaft (Bonn) within the network CRC TR 183.

References

- [1] M. Reiher, N. Wiebe, K. M. Svore, D. Wecker, and M. Troyer, *Elucidating reaction mechanisms on quantum computers*, *PNAS* **114**, 7555 (2017).
- [2] R. Babbush, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, A. Paler, A. Fowler, and H. Neven, *Encoding electronic spectra in quantum circuits with linear T complexity*, *Phys. Rev. X* **8**, 041015 (2018).
- [3] J. Preskill, *Reliable quantum computers*, *Proc. Roy. Soc. Lond. A* **454**, 385 (1998).
- [4] B. M. Terhal, *Quantum error correction for quantum memories*, *Rev. Mod. Phys.* **87**, 307 (2015).
- [5] E. T. Campbell, B. M. Terhal, and C. Vuillot, *Roads towards fault-tolerant universal quantum computation*, *Nature* **549**, 172 (2017).
- [6] A. Y. Kitaev, *Fault-tolerant quantum computation by anyons*, *Ann. Phys.* **303**, 2 (2003).
- [7] 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).
- [8] H. Bombin, *Topological order with a twist: Ising anyons from an abelian model*, *Phys. Rev. Lett.* **105**, 030403 (2010).
- [9] C. Horsman, A. G. Fowler, S. Devitt, and R. V. Meter, *Surface code quantum computing by lattice surgery*, *New J. Phys.* **14**, 123011 (2012).
- [10] B. J. Brown, K. Laubscher, M. S. Kesselring, and J. R. Wootton, *Poking holes and cutting corners to achieve Clifford gates with the surface code*, *Phys. Rev. X* **7**, 021029 (2017).
- [11] D. Litinski and F. v. Oppen, *Lattice Surgery with a Twist: Simplifying Clifford Gates of Surface Codes*, *Quantum* **2**, 62 (2018).
- [12] A. G. Fowler and C. Gidney, *Low overhead quantum computation using lattice surgery*, [arXiv:1808.06709](https://arxiv.org/abs/1808.06709) (2018).
- [13] A. J. Landahl and C. Ryan-Anderson, *Quantum computing by color-code lattice surgery*, [arXiv:1407.5103](https://arxiv.org/abs/1407.5103) (2014).
- [14] Y. Li, *A magic state’s fidelity can be superior to the*

- operations that created it, *New J. Phys.* **17**, 023037 (2015).
- [15] D. Herr, F. Nori, and S. J. Devitt, *Optimization of lattice surgery is NP-hard*, *npj Quant. Inf.* **3**, 35 (2017).
- [16] S. Bravyi and A. Kitaev, *Universal quantum computation with ideal Clifford gates and noisy ancillas*, *Phys. Rev. A* **71**, 022316 (2005).
- [17] J. Haah and M. B. Hastings, *Codes and Protocols for Distilling T, controlled-S, and Toffoli Gates*, *Quantum* **2**, 71 (2018).
- [18] S. Bravyi and J. Haah, *Magic-state distillation with low overhead*, *Phys. Rev. A* **86**, 052329 (2012).
- [19] C. Jones, *Multilevel distillation of magic states for quantum computing*, *Phys. Rev. A* **87**, 042305 (2013).
- [20] A. G. Fowler, S. J. Devitt, and C. Jones, *Surface code implementation of block code state distillation*, *Scientific Rep.* **3**, 1939 (2013).
- [21] A. G. Fowler, *Time-optimal quantum computation*, [arXiv:1210.4626](https://arxiv.org/abs/1210.4626) (2012).
- [22] D. Gottesman, *The Heisenberg representation of quantum computers*, *Proc. XXII Int. Coll. Group. Th. Meth. Phys.* **1**, 32 (1999).
- [23] V. Kliuchnikov, D. Maslov, and M. Mosca, *Fast and efficient exact synthesis of single-qubit unitaries generated by Clifford and T gates*, *Quantum Info. Comput.* **13**, 607 (2013).
- [24] V. Kliuchnikov, D. Maslov, and M. Mosca, *Asymptotically optimal approximation of single qubit unitaries by Clifford and T circuits using a constant number of ancillary qubits*, *Phys. Rev. Lett.* **110**, 190502 (2013).
- [25] D. Gosset, V. Kliuchnikov, M. Mosca, and V. Russo, *An algorithm for the T-count*, [arXiv:1308.4134](https://arxiv.org/abs/1308.4134) (2013).
- [26] L. E. Heyfron and E. T. Campbell, *An efficient quantum compiler that reduces T count*, *Quantum Sci. Technol.* **4**, 015004 (2018).
- [27] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, *A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits*, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **32**, 818 (2013).
- [28] P. Selinger, *Quantum circuits of T-depth one*, *Phys. Rev. A* **87**, 042302 (2013).
- [29] M. Amy, D. Maslov, and M. Mosca, *Polynomial-time T-depth optimization of Clifford+T circuits via matroid partitioning*, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **33**, 1476 (2014).
- [30] D. Litinski and F. von Oppen, *Quantum computing with Majorana fermion codes*, *Phys. Rev. B* **97**, 205404 (2018).
- [31] A. Lavasani and M. Barkeshli, *Low overhead Clifford gates from joint measurements in surface, color, and hyperbolic codes*, *Phys. Rev. A* **98**, 052319 (2018).
- [32] J. I. Hall, Notes on Coding Theory Chapter 6: Modifying Codes, <https://users.math.msu.edu/users/jhall/classes/codenotes/Mod.pdf>, accessed: 2019-01-30.
- [33] E. T. Campbell and M. Howard, *Magic state parity-checker with pre-distilled components*, *Quantum* **2**, 56 (2018).
- [34] A. M. Meier, B. Eastin, and E. Knill, *Magic-state distillation with the four-qubit code*, *Quant. Inf. Comp.* **13**, 195 (2013).
- [35] E. T. Campbell and J. O’Gorman, *An efficient magic state approach to small angle rotations*, *Quantum Sci. Technol.* **1**, 015007 (2016).
- [36] D. Herr, F. Nori, and S. J. Devitt, *Lattice surgery translation for quantum computation*, *New J. Phys.* **19**, 013034 (2017).
- [37] A. G. Fowler and S. J. Devitt, *A bridge to lower overhead quantum computation*, [arXiv:1209.0510](https://arxiv.org/abs/1209.0510) (2012).
- [38] C. Gidney and A. G. Fowler, *Efficient magic state factories with a catalyzed $|CCZ\rangle$ to $2|T\rangle$ transformation*, [arXiv:1812.01238](https://arxiv.org/abs/1812.01238) (2018).
- [39] C. H. Bennett, G. Brassard, S. Popescu, B. Schumacher, J. A. Smolin, and W. K. Wootters, *Purification of noisy entanglement and faithful teleportation via noisy channels*, *Phys. Rev. Lett.* **76**, 722 (1996).
- [40] C. H. Bennett, H. J. Bernstein, S. Popescu, and B. Schumacher, *Concentrating partial entanglement by local operations*, *Phys. Rev. A* **53**, 2046 (1996).
- [41] C. Dickel, J. J. Westorp, N. K. Langford, S. Peiter, R. Sagastizabal, A. Bruno, B. Criger, F. Motzoi, and L. DiCarlo, *Chip-to-chip entanglement of transmon qubits using engineered measurement fields*, *Phys. Rev. B* **97**, 064508 (2018).
- [42] P. Campagne-Ibarcq, E. Zalys-Geller, A. Narla, S. Shankar, P. Reinhold, L. Burkhardt, C. Axline, W. Pfaff, L. Frunzio, R. J. Schoelkopf, and M. H. Devoret, *Deterministic remote entanglement of superconducting circuits through microwave two-photon transitions*, *Phys. Rev. Lett.* **120**, 200501 (2018).
- [43] C. J. Axline, L. D. Burkhardt, W. Pfaff, M. Zhang, K. Chou, P. Campagne-Ibarcq, P. Reinhold, L. Frunzio, S. Girvin, L. Jiang, *et al.*, *On-demand quantum state transfer and entanglement between remote microwave cavity memories*, *Nat. Phys.* **14**, 705 (2018).

- [44] N. J. Ross and P. Selinger, *Optimal ancilla-free Clifford+T approximation of z-rotations*, arXiv:1403.2975 (2014).
- [45] G. Duclos-Cianci and D. Poulin, *Reducing the quantum-computing overhead with complex gate distillation*, *Phys. Rev. A* **91**, 042315 (2015).
- [46] A. W. Harrow, B. Recht, and I. L. Chuang, *Efficient discrete approximations of quantum gates*, *Journal of Mathematical Physics* **43**, 4445 (2002).
- [47] G. Duclos-Cianci and K. M. Svore, *Distillation of nonstabilizer states for universal quantum computation*, *Phys. Rev. A* **88**, 042325 (2013).
- [48] A. Bocharov, Y. Gurevich, and K. M. Svore, *Efficient decomposition of single-qubit gates into v basis circuits*, *Phys. Rev. A* **88**, 012313 (2013).
- [49] N. C. Jones, J. D. Whitfield, P. L. McMahon, M.-H. Yung, R. V. Meter, A. Aspuru-Guzik, and Y. Yamamoto, *Faster quantum chemistry simulation on fault-tolerant quantum computers*, *New J. Phys.* **14**, 115023 (2012).
- [50] G. H. Low and I. L. Chuang, *Hamiltonian simulation by qubitization*, arXiv:1610.06546 (2016).
- [51] G. H. Low and I. L. Chuang, *Optimal Hamiltonian simulation by quantum signal processing*, *Phys. Rev. Lett.* **118**, 010501 (2017).
- [52] R. Babbush, D. W. Berry, J. R. McClean, and H. Neven, *Quantum simulation of chemistry with sublinear scaling to the continuum*, arXiv:1807.09802 (2018).
- [53] C. Jones, *Low-overhead constructions for the fault-tolerant Toffoli gate*, *Phys. Rev. A* **87**, 022328 (2013).
- [54] C. Gidney, *Halving the cost of quantum addition*, *Quantum* **2**, 74 (2018).
- [55] E. T. Campbell and M. Howard, *Unified framework for magic state distillation and multiqubit gate synthesis with reduced resource cost*, *Phys. Rev. A* **95**, 022316 (2017).
- [56] J. O’Gorman and E. T. Campbell, *Quantum computation with realistic magic-state factories*, *Phys. Rev. A* **95**, 032338 (2017).
- [57] K. K. Likharev and V. K. Semenov, *RSFQ logic/memory family: A new Josephson-junction technology for sub-terahertz-clock-frequency digital systems*, *IEEE Transactions on Applied Superconductivity* **1**, 3 (1991).
- [58] A. G. Fowler, S. J. Devitt, and C. Jones, *Synthesis of arbitrary quantum circuits to topological assembly: Systematic, online and compact*, *Scientific Rep.* **7**, 10414 (2017).
- [59] A. Paler, I. Polian, K. Nemoto, and S. J. Devitt, *Fault-tolerant, high-level quantum circuits: form, compilation and description*, *Quantum Sci. Technol.* **2**, 025003 (2017).
- [60] L. Lao, B. van Wee, I. Ashraf, J. van Someren, N. Khammassi, K. Bertels, and C. G. Almudever, *Mapping of lattice surgery-based quantum circuits on surface code architectures*, *Quantum Sci. Technol.* **4**, 015005 (2018).
- [61] H. Bombin and M. A. Martin-Delgado, *Topological quantum distillation*, *Phys. Rev. Lett.* **97**, 180501 (2006).
- [62] M. S. Kesselring, F. Pastawski, J. Eisert, and B. J. Brown, *The boundaries and twist defects of the color code and their applications to topological quantum computation*, *Quantum* **2**, 101 (2018).
- [63] H. P. Nautrup, N. Friis, and H. J. Briegel, *Fault-tolerant interface between quantum memories and quantum processors*, *Nat. Commun.* **8**, 1321 (2017).
- [64] D. Litinski and F. von Oppen, *Braiding by Majorana tracking and long-range CNOT gates with color codes*, *Phys. Rev. B* **96**, 205413 (2017).
- [65] IBM doubling qubits every 8 months, <https://www.nextbigfuture.com/2018/02/ibm-doubling-qubits-every-8-months-and-ecommerce-cryptography-at-risk-in-7-15-years.html>, accessed: 2018-08-01.

A Surface-code qubits and lattice-surgery operations

To illustrate the translation of protocols in our framework into surface-code patches, we show how the patches of Fig. 1 and the rules of the game and protocols of Fig. 2 are implemented with surface codes.

Surface-code patches. Each patch corresponds to a surface-code patch with code distance d . Therefore, each tile corresponds to d^2 physical data qubits, as shown in Fig. 37 for $d = 5$. In our surface-code patches,

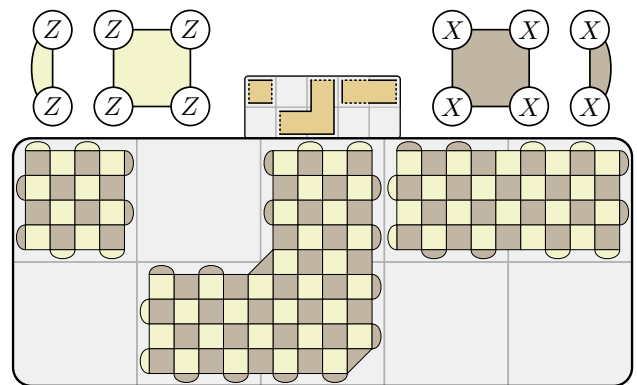


Figure 37: Surface-code implementation of the patches shown in Fig. 1. Physical qubits are placed on vertices. Bright faces correspond to Z stabilizers and dark faces to X stabilizers.

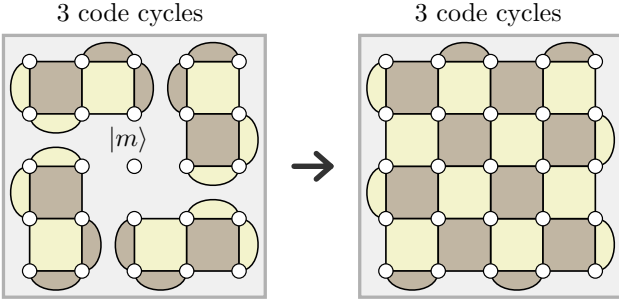


Figure 38: State-injection protocol of Ref. [13].

physical qubits are placed on the vertices, bright faces correspond to Z stabilizers and dark faces to X stabilizers. Solid and dashed boundaries correspond to X and Z boundaries (also called rough and smooth boundaries). For one-qubit patches, the product of all d physical X (Z) operators along any of the X (Z) boundaries is the logical X (Z) operator of the encoded qubit. For two-qubit patches with six boundaries, the string operators located at the boundaries correspond to the logical operators shown in Fig. 1, i.e., going clockwise, X_1 , Z_1 , $X_1 \cdot X_2$, Z_2 , X_2 , and $Z_1 \cdot Z_2$. Note that, in principle, the width of two-tile patches can be $2d - 1$ instead of $2d$, potentially reducing the space cost [11]. Furthermore, the correspondence between solid and dashed, and X and Z boundaries is interchangeable.

State initialization. We now show how the operations and protocols of Fig. 2 are implemented with surface codes for $d = 5$, and motivate their time cost in the framework, where the reasoning is that $1 \oplus$ is associated with operations whose time cost scales with d . Surface-code patches can be initialized in the logical $|0\rangle$ or $|+\rangle$ state by initializing all physical qubits of the patch in $|0\rangle$ or $|+\rangle$, and then measuring all stabilizers.

Naively, one would expect that there should be a time cost associated with this operation, since the stabilizers need to be measured for d code cycles to account for measurement errors. However, this can be done simultaneously with the subsequent lattice-surgery operation, as will become apparent in the example of the Bell state preparation. For arbitrary states, the logical states are prepared via state injection. This is a non-fault-tolerant procedure with a constant time cost that does not scale with d , which is why we do not associate a time step with it. One such state-injection protocol is described in Ref. [13] and is shown in Fig. 38 for the preparation of a logical magic state $|m\rangle$. In the left panel, a physical magic state is prepared, along with a stabilizer state by measuring the shown stabilizers for three code cycles. Note that any single-qubit error during these three code cycles will corrupt the logical information. Next, the stabilizer configuration is switched to the or-

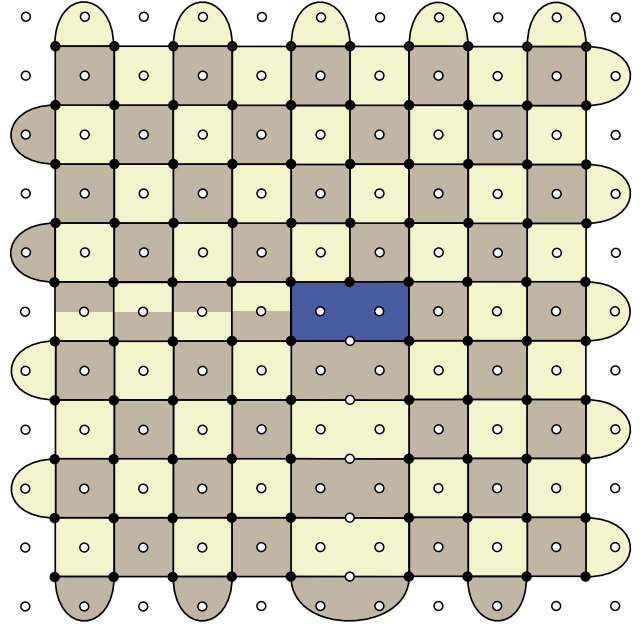


Figure 39: Twist-based lattice surgery in a square lattice of qubits with nearest-neighbor couplings. The black dots are physical data qubits and the white dots are physical measurement qubits.

dinary surface code in the right panel. Here, the stabilizers are, again, only measured for three code cycles, independently of d , since the state-injection protocol is, in any case, non-fault-tolerant, i.e., produces logical states with an error rate proportional to the physical error rate p .

Patch measurement and Bell state preparation. Surface-code patches are measured in the X or Z basis by measuring all physical qubits in the corresponding basis and performing some classical error correction, where the time cost does not scale with d . Two-patch measurements correspond to lattice surgery and can be demonstrated via the preparation of a Bell state, as shown in Fig. 40a. Two surface-code patches are initialized in the logical $|+\rangle$ state by initializing all physical qubits in $|+\rangle$ and measuring the stabilizers. Simultaneously, lattice surgery between the two patches is performed, measuring the logical $Z \otimes Z$ operator. The measurement outcome is the product of the newly introduced Z stabilizers highlighted in red, as the product of these stabilizers corresponds to the product of the logical Z operators encoded in the two surface-code Z boundaries. To account for measurement errors, this measurement is repeated for d code cycles. Finally, the patch is split into two patches again, leaving the two logical surface-code qubits in an entangled Bell state.

Y measurements. Two-patch measurements can be used to measure products of two Pauli operators other

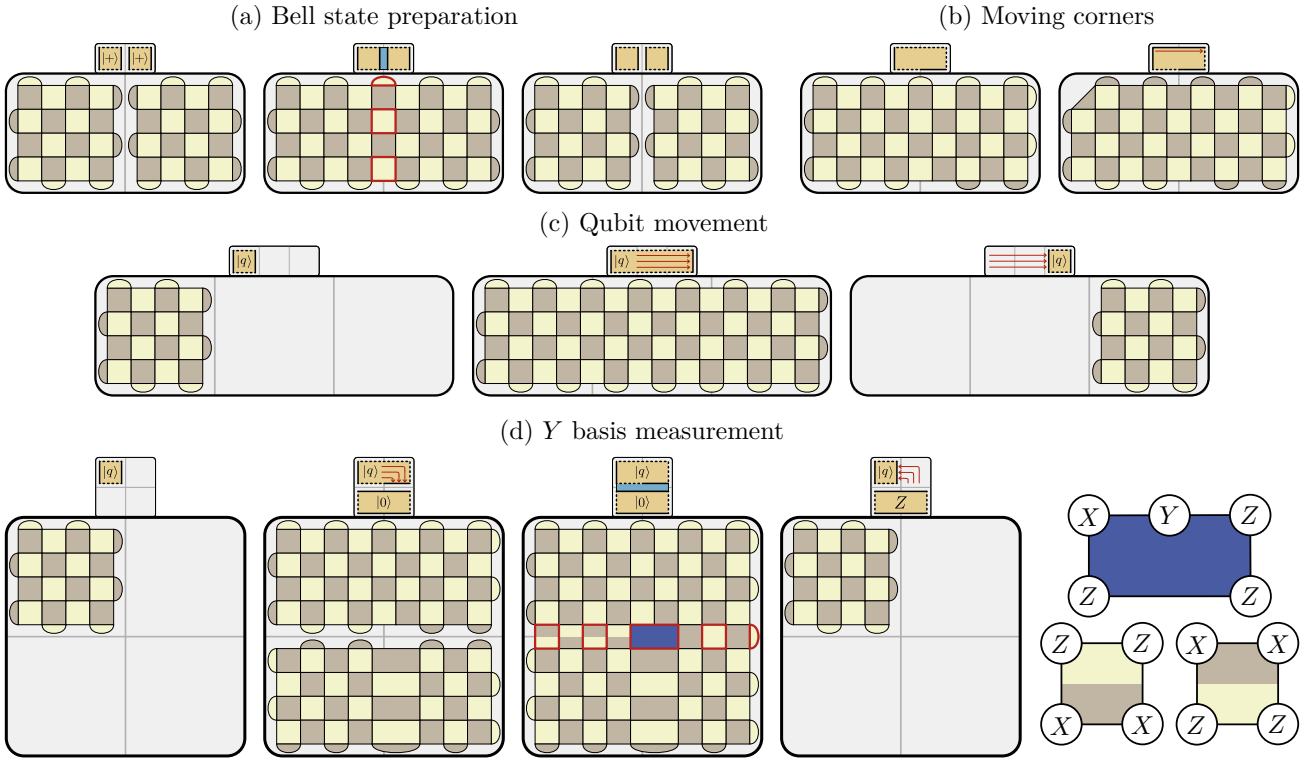


Figure 40: Surface-code implementation of the protocols in Fig. 2a-d.

than $Z \otimes Z$, e.g., operators involving the Y operator, as shown in Fig. 40d. First, a patch is deformed to a wider patch by initializing physical qubits in the X basis and measuring the new stabilizers, which takes d code cycles. Below the wide patch, a rectangular ancilla patch is initialized in the $|0\rangle$ state. A column of physical qubits in the center is missing, so that, in the next step, the ancilla can be used for twist-based lattice surgery [11], measuring the Y operator. The product of the operators highlighted in red in the third step corresponds to the logical $Y \otimes Z$ operator between the two logical qubits. The lattice surgery in the third step involves dislocation operators and a five-qubit twist defect. Even though these stabilizers are irregular, they can still be measured in a square lattice of physical qubits with nearest-neighbor couplings, as we show in Fig. 39. For the measurement of twist operators and wide X and Z stabilizers, up to three measurement ancillas can be used.

Multi-patch measurements. For a multi-patch measurement in Fig. 41, all physical qubits located in the region of the ancilla patch are initialized in the $|+\rangle$ state. Next, new check operators are introduced. The newly introduced X -type stabilizers all yield trivial outcomes, since they are products of physical qubits initialized in an X eigenstate and previously measured check operators. The nontrivial operators are highlighted by

a red dot in Fig. 41. Their product is equivalent to the desired operator, i.e., $Y_{|q_1\rangle} \otimes X_{|q_3\rangle} \otimes Z_{|q_4\rangle} \otimes X_{|q_5\rangle}$. The new check operators are measured for d code cycles to account for measurement errors. This procedure corresponds to the multi-body lattice surgery protocol introduced in Ref. [12]. It can be used to measure any product of surface-code-boundary Pauli operators by initializing physical qubits in the $|+\rangle$ state in an ancilla region of width d , and then measuring new check operators, where the product of the nontrivial operators yields the outcome of the desired multi-patch measurement. The ancilla region of width d is required to ensure that the code distance of the stabilizer configuration during the multi-body lattice surgery remains d .

Moving boundaries. The protocol to move patches is similar to lattice surgery. It is shown in Fig. 40c. Extending the patch via its Z boundary in the second step is the same operation as a $Z \otimes Z$ lattice surgery between the patch and a rectangular $|+\rangle$ ancilla qubit to the right. This needs to be done for d code cycles to account for measurement errors. Finally, the patch is shortened again by measuring the left two thirds of physical qubits in the X basis.

Moving corners. The movement of corners of a surface-code patch is shown in Fig. 40b. It corresponds to a change of boundary stabilizers. In order to account for measurement errors of the newly measured stabiliz-

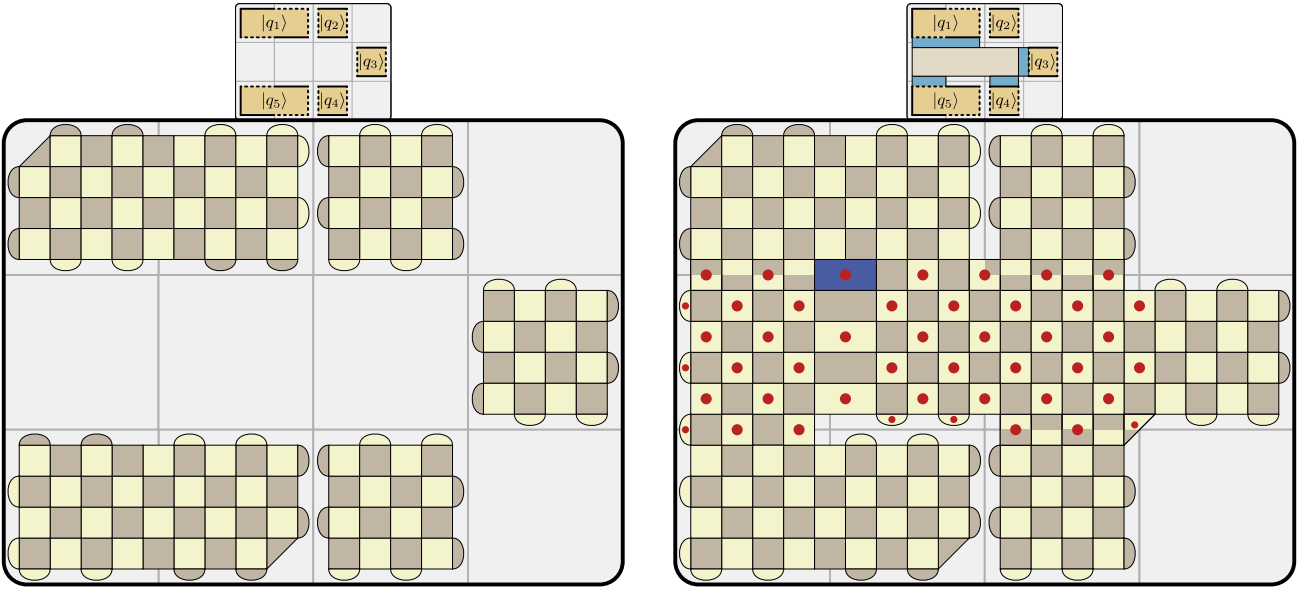


Figure 41: Surface-code implementation of the multi-patch measurement in Fig. 2e. The measurement outcome is the product of all check operators with a red dot.

ers, this requires d code cycles. The top left physical qubit in the second step of Fig. 40b is removed from the patch via an X measurement.

B Extended ruleset

Some surface-code operations are not covered by the rules discussed in the introduction. In particular, we only consider patches with 4 or 6 corners, where we refer to the points where two edges meet as corners. In general, one could also consider patches with a

higher number of corners. A patch with $2N + 2$ corners represents N qubits, as shown in Fig. 42. The simplest case is a four-corner patch (a/b) representing a single qubit. Six-corner patches (c) are two-qubit patches. The general rule that assigns the operators of N qubits to the edges of a $(2N + 2)$ -corner patch is given in Fig. 42d. Going clockwise, the dashed boundaries correspond to $X_1, X_1X_2, X_2X_3, \dots, X_{N-1}X_N$ and X_N . Starting to the right of X_1 , the solid edges correspond to Z_1, Z_2, \dots, Z_N and the product $Z_1Z_2 \cdots Z_N$.

One can also consider patches with shortened edges, such that they occupy fewer tiles. The drawback of this is that in every time step, an error corresponding to the Pauli operator represented by the shortened edge will occur with a certain probability p_{err} . An example of a six-corner patch with two shortened X edges is shown in Fig. 43, meaning that this six-corner patch is susceptible to X errors. In the surface-code implementation, this corresponds to a patch with boundaries that are shorter than d physical data qubits, effectively reducing the code distance of the logical operators encoded by the shortened edges. Note that patches with shortened edges may occupy more than d^2 physical data qubits per tile.

With $(2N + 2)$ -corner patches, the set of operations needs to be modified. The initialization rule for such patches is:

- Qubits can be initialized in the X and Z eigenstates $|+\rangle$ and $|0\rangle$. All qubits that are part of one patch must be initialized in the same state. (Cost: $0\oplus$)

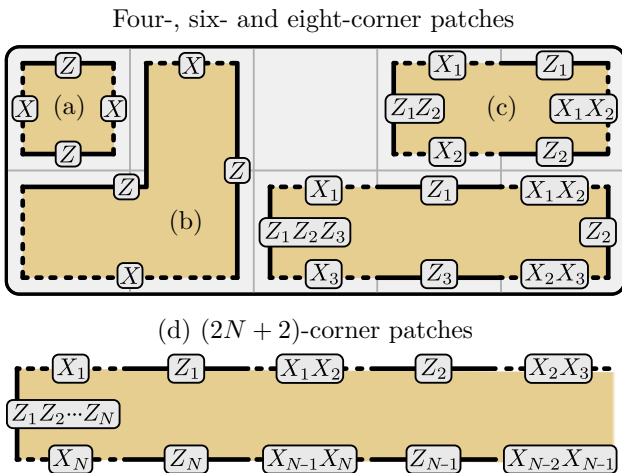


Figure 42: Patches with $2N + 2$ corners represent N qubits. Their $2N + 2$ edges represent the shown Pauli operators.

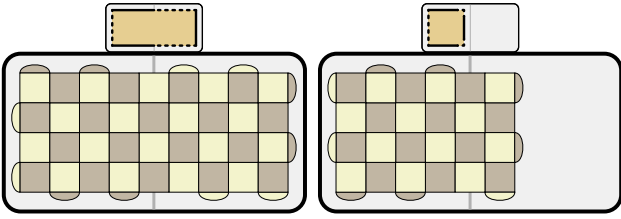


Figure 43: Surface-code implementation of a six-corner patch with shortened boundaries

Similarly, the single-patch measurement rule is modified to

- Qubits can be measured in the X or Z basis. All qubits that are part of the same patch are measured simultaneously and in the same basis. This measurement removes the patch from the board. (Cost: $0\ominus$)

Pauli product measurements. Using multi-corner patches with shortened boundaries, the multi-patch measurement rule is, in principle, redundant. For instance, the Pauli product measurement of Fig. 8 can be equivalently performed in $1\ominus$ via the protocol shown in Fig. 44. An 8-corner ancilla patch is initialized in the $|+\rangle^{\otimes 3}$ state. The shape of this patch is chosen, such that each of the four Z edges is adjacent to one of the four operators that are part of the measurement. Note that this means that some of the X edges are shortened, such that the qubits are susceptible to X errors. In this case, this is not a problem, since the qubits are initialized in X eigenstates and random X errors will cause no change to the states. Next, in step 3, we measure the four Pauli products $Z_{|q_1\rangle} \otimes Z_1$, $Y_{|q_2\rangle} \otimes Z_2$, $Z_{|m\rangle} \otimes Z_3$ and $X_{|q_4\rangle} \otimes (Z_1 \cdot Z_2 \cdot Z_3)$. Because the ancilla is initialized in an X eigenstate, the operators Z_1 , Z_2 and Z_3 are unknown, and the outcome of each of the four aforementioned measurements is entirely random. However, multiplying the four measurement outcomes yields $Z_{|q_1\rangle} \otimes Y_{|q_2\rangle} \otimes X_{|q_4\rangle} \otimes Z_{|m\rangle} \otimes (Z_1 \cdot Z_2 \cdot Z_3 \cdot Z_1 \cdot Z_2 \cdot Z_3)$, which is precisely the operator $Z_{|q_1\rangle} \otimes Y_{|q_2\rangle} \otimes X_{|q_4\rangle} \otimes Z_{|m\rangle}$ that we wanted to measure. Finally, to discard the ancilla patch we measure its three qubits in the X basis. Again, X errors will have no effect, as they commute with the measurement basis. Measurement outcomes of $X_i = -1$ prompt a Pauli correction. If in the previous step, the Z_i edge was measured together with a Pauli operator P , the correction is a $P_{\pi/2}$ gate. For instance, if in Fig. 8 the final measurements yield $X_2 = -1$ and $X_3 = -1$, the corrections are a $Y_{\pi/2}$ rotation on $|q_2\rangle$ and a $Z_{\pi/2}$ rotation on $|m\rangle$.

This type of protocol can be used to measure any product of n Pauli operators. An ancilla patch needs to be initialized in the $|+\rangle^{\otimes n}$ state with Z edges adja-

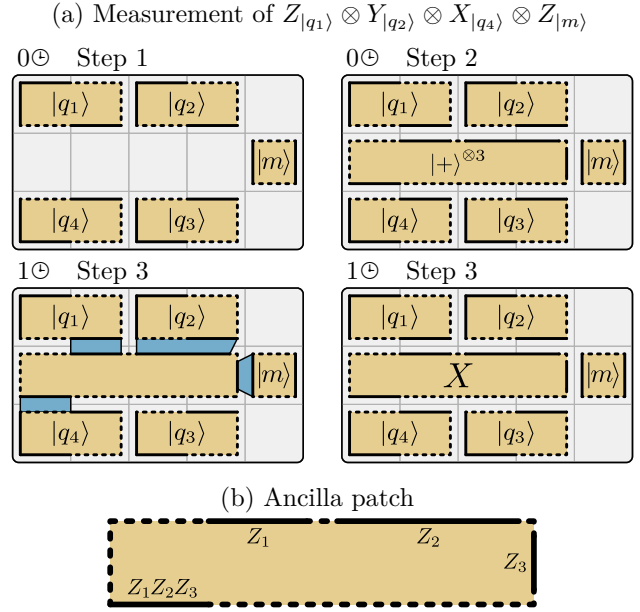


Figure 44: Pauli product measurement protocol. (a) Example of a measurement of the operator $Z \otimes Y \otimes \mathbb{1} \otimes X \otimes Z$ of the qubits $|q_1\rangle$, $|q_2\rangle$, $|q_3\rangle$, $|q_4\rangle$ and $|m\rangle$. (b) Ancilla patch used during the measurement.

cent to the n operators part of the measurement. The surface-code implementation of this protocol is identical to the surface-code implementation of multi-patch measurements in Fig. 41.

While multi-corner patches and shortened edges increase the number of surface-code operations that are covered by the framework, there are still rules that can be added to the ruleset to account for more operations, such as, e.g., the movement of corners inside a patch [10]. Also, for the initialization of non-Pauli eigenstates, error models other than random Pauli errors can be considered.

C Proof-of-principle device

Here, we discuss how $(3d - 1) \cdot 2d$ physical data qubits can be used to build a proof-of-principle device that is a universal two-qubit error-corrected quantum computer that uses undistilled magic states and can demonstrate all the operations required for large-scale quantum computing. We go through the example of a computation that starts with three $\pi/8$ rotations around $Z \otimes Z$, $Y \otimes X$ and $Y \otimes Y$ in Fig. 45. For the first rotation, we need to measure $Z_1 \otimes Z_2 \otimes Z_{|m\rangle}$. A magic state is initialized in a long patch in step 2, which is equivalent to initializing a magic state and measuring $X \otimes X$ between the magic state and neighboring $|0\rangle$ ancillas. This effectively en-

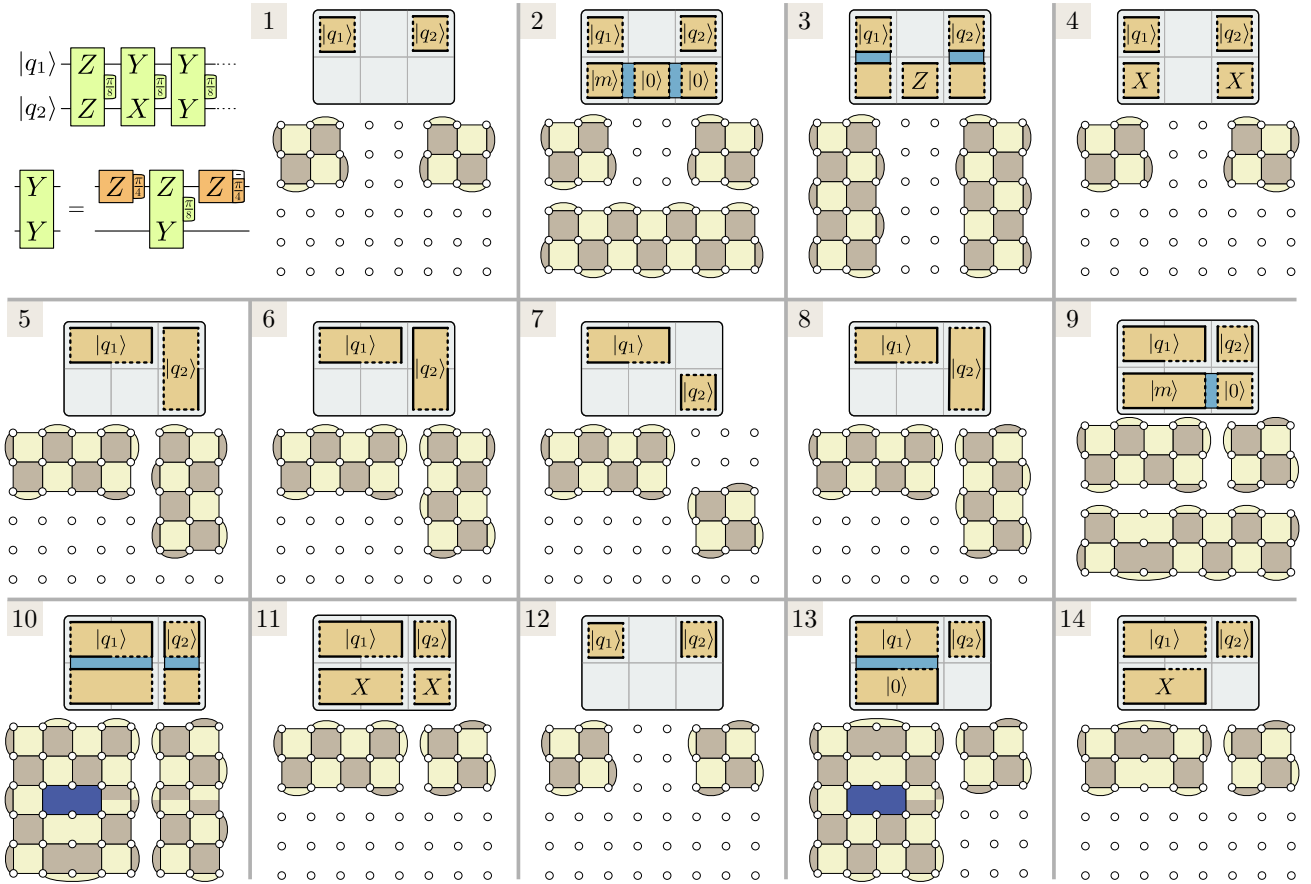


Figure 45: Proof-of-principle two-qubit device implemented with 48 physical data qubits.

codes the magic state in a three-qubit repetition code with a logical Z operator $Z_L = Z \otimes Z \otimes Z$. To consume the magic state, $Z_1 \otimes Z_2 \otimes Z_L$ is measured in step 3. This consumes a magic state for the $Z \otimes Z$ rotation.

The next rotation is a $Y \otimes X$ rotation. Here, we first need to deform $|q_1\rangle$, such that both the X and Z boundaries of the qubit are accessible. Qubit $|q_2\rangle$ is rotated in steps 5-8 using the protocol in Fig. 11a. In step 9, again, a magic state is initialized in a two-qubit repetition code with $Z_L = Z_{a1} \otimes Z_{a2}$. In step 10, the magic state is consumed via a $Y_1 \otimes Z_{a1}$ and a $X_1 \otimes Z_{a2}$ measurement.

This kind of protocol consisting of patch deformations and patch rotations can be used to perform any $\pi/8$ rotation with the exception of $(Y \otimes Y)_{\pi/8}$, since there is not enough space to make both Y operators accessible for lattice surgery. For this rotation, we first explicitly execute a Clifford gate to change $(Y \otimes Y)_{\pi/8}$ to any other rotation. Any Clifford gate that does not commute with $Y \otimes Y$ will suffice. In our example, we choose a $Z_{\pi/4}$ rotation. It is performed by initializing a $|0\rangle$ state in step 13, and measuring $Z_1 \otimes Y$ between $|q_1\rangle$ and the

ancilla, following the protocol of Fig. 11b.

This demonstrates that a proof-of-principle experiment can be built with 48 physical data qubits. In general, this requires $6d^2 - 2d$ qubits, i.e., 48 for $d = 3$, 140 for $d = 5$ and 280 for $d = 7$. If measurement qubits are required for syndrome readout, the number of physical qubits roughly doubles.

D Implementation of the 7-to-1 protocol

Even though the distillation of $|Y\rangle = |0\rangle + i|1\rangle$ states has no use in our framework, we show how to implement the 7-to-1 distillation protocol for benchmarking purposes in Fig. 46. The protocol is based on the 7-qubit Steane code. Its X stabilizers are the faces shown in Fig. 46a, and its logical X operator can be chosen as the $X \otimes X \otimes X$ operator with support on the three qubits drawn in red.

Following the procedure in Sec. 3, the distillation circuit is obtained by initializing $m_x + k = 4$ qubits in

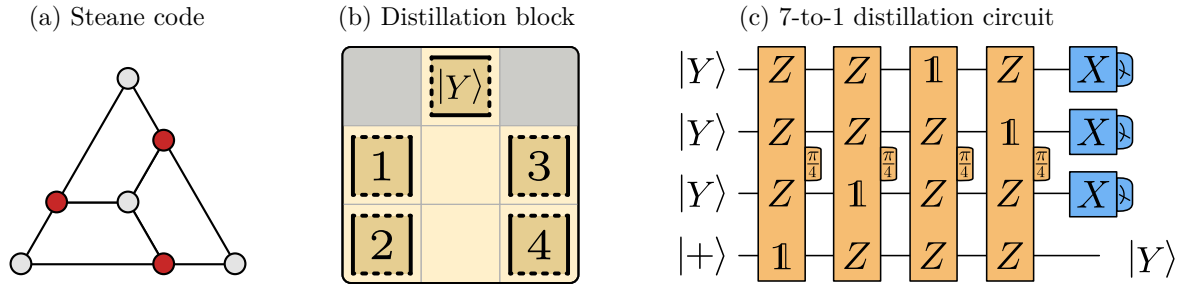


Figure 46: The Steane code (a) is the basis of 7-to-1 distillation (c). In our framework, the corresponding distillation block (b) uses 7 tiles for $4\ominus$.

the $|+\rangle$ state, where the first three qubits are associated with the three X stabilizers, and the last qubit is associated with the logical X operator. For each qubit of the Steane code, the circuit contains a $\pi/4$ rotation with Z 's on each stabilizer and logical operator that the qubit is part of. The three qubits in the corner of the triangle are only part of a single stabilizer and no logical operator, therefore they contribute with single-qubit $Z_{\pi/4}$ rotations, which can be absorbed into

the initial state. The remaining four rotations are shown in Fig. 46c.

A distillation block that can be used for this protocol is shown in Fig. 46b. Since the consumption of $|Y\rangle$ resource states requires no Clifford correction, this block consists of only 7 tiles. With four rotations, the leading order of the space-time cost of this protocol is $7d^2 \cdot 4d = 28d^3$.

2 | Lattice Surgery with a Twist

One of the techniques used in the previous chapter for logical Pauli product measurements between surface-code patches was twist-based lattice surgery, which can measure products involving the Y operator. While twist-based lattice surgery was briefly outlined in the appendix of the previous publication, it is properly introduced and analyzed in the following publication. We also show how the required stabilizer measurements can be implemented in physical architectures using solid-state qubits.

Lattice Surgery with a Twist: Simplifying Clifford Gates of Surface Codes

Daniel Litinski and Felix von Oppen

Dahlem Center for Complex Quantum Systems and Fachbereich Physik, Freie Universität Berlin, Arnimallee 14, 14195 Berlin, Germany

We present a planar surface-code-based scheme for fault-tolerant quantum computation which eliminates the time overhead of single-qubit Clifford gates, and implements long-range multi-target CNOT gates with a time overhead that scales only logarithmically with the control-target separation. This is done by replacing hardware operations for single-qubit Clifford gates with a classical tracking protocol. Inter-qubit communication is added via a modified lattice surgery protocol that employs twist defects of the surface code. The long-range multi-target CNOT gates facilitate magic state distillation, which renders our scheme fault-tolerant and universal.

1 Introduction

The performance of quantum computers is limited by the coherence times of the underlying physical qubits. Quantum error correction [1] offers the possibility to enhance the qubits' survival times by encoding quantum information using logical qubits consisting of many physical qubits. Topological quantum error-correcting codes [2, 3] are of particular interest, as they only require the measurement of spatially local operators – a feature that is compatible with the local operations accessible in two-dimensional solid-state qubit architectures, such as superconducting qubits [4], spin qubits [5], or Majorana-based qubits [6].

Quantum error-correcting codes typically operate in cycles. In each code cycle, mutually commuting operators called stabilizers [7] are measured to reveal the error syndrome, which is used to determine and correct errors. Surface codes [8, 9] are topological codes that feature a high error threshold [10, 11], and only require the measurement of four-qubit stabilizer operators for the readout of the error syndrome. The low-weight stabilizers are an advantage over other codes such as color codes [12, 13], which require the measurement of six-qubit operators. This facilitates syndrome readout in many physical architectures such as superconducting qubits, where the measurement of higher-weight sta-

bilizers requires more potentially faulty controlled-not (CNOT) gates.

The main drawback of surface codes in comparison to color codes is the absence of transversal single-qubit Clifford gates, i.e., the gates that are products of the Hadamard gate H and the phase gate S . While the transversal Clifford gates of color codes provide them with fast logical H and S gates, defect-based proposals for surface codes [14] implement the H gate via a multi-step measurement protocol, and the S gate via a distilled ancilla qubit. In order to lower the overhead of single-qubit Clifford gates, surface code qubits can be encoded using twist defects [15], which are essentially Majoranas that can be braided via code deformation [16]. It was pointed out that braiding of twists can also be implemented via a classical tracking protocol [17], in accordance with the Gottesman-Knill theorem [18].

In this work, we present a scheme that implements this tracking protocol for planar surface codes, as opposed to twist-based encodings. We refer to this protocol as *edge tracking*. In our scheme, Clifford completeness is achieved via a modified lattice surgery [19]

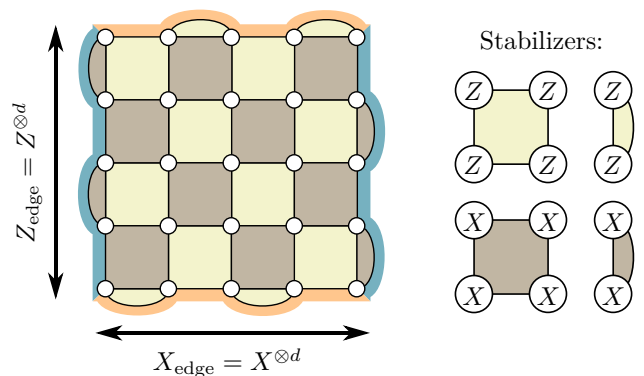


Figure 1: An example of a surface code qubit with code distance $d = 5$. Physical qubits are located on the vertices, and the faces define the two- and four-qubit Z type (bright) and X type (dark) stabilizer operators. X strings along the X edge (orange) are logical X_L operators, whereas Z strings along Z edges (blue) are Z_L operators.

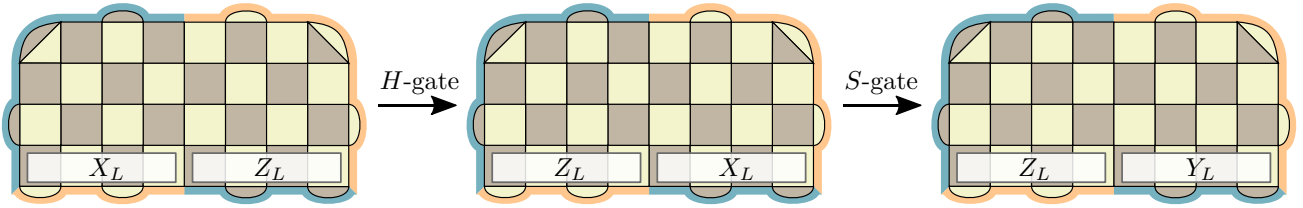


Figure 2: An example of edge tracking with a wide surface code qubit. Starting from the default encoding $X_{\text{edge}} = X_L$ and $Z_{\text{edge}} = Z_L$, an H gate changes it to $X_{\text{edge}} = Z_L$ and $Z_{\text{edge}} = X_L$, and a subsequent S gate modifies it to $X_{\text{edge}} = Z_L$ and $Z_{\text{edge}} = Y_L$.

protocol. Twist defects are no longer used to encode quantum information, but reappear in lattice-surgery protocols involving the logical Y_L operator, so that we refer to the protocol as *twist-based lattice surgery*. Our scheme provides long-range multi-target CNOT gates – i.e., CNOTs with one control and arbitrarily many targets – between any set of edge-tracked surface code qubits. These gates are particularly useful for magic state distillation [20], which completes the universal gate set by fault-tolerantly implementing the T gate (or $\pi/8$ gate). Our scheme not only eliminates the need for hardware operations for single-qubit Clifford gates, but also conceptually simplifies the twist-defect-based approach to surface-code quantum computing. Even though our scheme features twist defects and dislocation lines, the only concepts necessary to understand our scheme are the encoding of logical qubits and the measurement of two-qubit parity operators. We discuss the implementation of the single-qubit Clifford gates, CNOT gates, and T gates in Secs. 2, 3 and 4, respectively. In a concluding section, we discuss our scheme in the context of possible hardware implementations and in comparison to alternative topological codes.

2 Edge Tracking

The basic framework of our scheme are physical qubits arranged on a 2D square lattice which allow for the measurement of local stabilizer operators. Examples of possible physical realizations include superconducting qubits emulating stabilizer measurements using ancilla qubits and CNOT gates [14], or Majorana-based qubits using direct measurements of the stabilizers via Majorana fermion parity measurements [21]. A single surface code qubit can be defined using the checkered square shown in Fig. 1, where physical qubits are located at the vertices. We refer to the Pauli operators of the physical qubits as X , Y , and Z . The faces define the $X^{\otimes n}$ - and $Z^{\otimes n}$ -stabilizers of the code, where n is the number of qubits that are part of the face. The figure shows an example of a code with code distance $d = 5$, but this construction can be generalized to arbitrary

code distances.

Surface code qubits have two distinct types of boundaries, usually referred to as rough and smooth edges. Here, we call them X and Z edges in analogy to the logical Pauli operators X_L and Z_L that they encode. Surface code qubits can be easily initialized in the logical $+1$ -eigenstates $|0_L\rangle$ and $|+_L\rangle$ of Z_L and X_L by initializing all physical qubits in the corresponding physical states $|0\rangle$ and $|+\rangle$, measuring all stabilizers, and correcting the errors. Conversely, they can be read out in the X_L and Z_L basis by measuring all physical qubits in the X or Z basis, and performing classical error correction.

We define the operator X_{edge} (Z_{edge}) as the string of X operators (Z operators) on all physical qubits along an X edge (Z edge). In the default encoding, $X_{\text{edge}} = X_L$ and $Z_{\text{edge}} = Z_L$. The edge tracking procedure that we now introduce essentially modifies which logical operators are encoded by X_{edge} and Z_{edge} . Logical single-qubit Clifford gates map the logical Pauli operators X_L , Y_L , and Z_L onto other Pauli operators. In particular, an H gate maps $X_L \rightarrow Z_L$, $Y_L \rightarrow -Y_L$, and $Z_L \rightarrow X_L$. An S gate maps $X_L \rightarrow Y_L$, $Y_L \rightarrow -X_L$, and $Z_L \rightarrow Z_L$. Thus, we can replace single-qubit Clifford gates by a classical tracking procedure. This is essentially the content of the Gottesman-Knill theorem [18], which states that Clifford gates can be simulated efficiently on a classical computer. For now, we only consider tracking of single-qubit Clifford gates H and S , whereas CNOT gates are performed explicitly.

In order to combine this tracking scheme with lattice surgery, it will be convenient to use the wide qubits shown in Fig. 2 instead of the square qubits that were previously introduced. These qubits have an X and Z edge on the same side, such that the logical operators X_L , Y_L and Z_L can all be accessed by lattice surgery from the same side of the qubit. Compared to square qubits with the same code distance, this comes at the price of a larger number of physical qubits for each logical qubit. The figure also shows an example of edge tracking. The default encoding is $X_{\text{edge}} = X_L$ and $Z_{\text{edge}} = Z_L$. An H gate changes the encoding to $X_{\text{edge}} = Z_L$ and

$Z_{\text{edge}} = X_L$. A subsequent S gate modifies it to $X_{\text{edge}} = Z_L$ and $Z_{\text{edge}} = Y_L$.

3 Lattice surgery with a twist

Edge tracking requires a suitable CNOT gate protocol in order to be useful for universal quantum computation. This is provided by twist-based lattice surgery. It essentially implements the circuit identity shown in Fig. 3 for edge-tracked qubits. Here, a CNOT between a control and target qubit corresponds to three measurements: a $Z \otimes Z$ parity measurement between the control and an ancilla initialized in the X eigenstate $|+\rangle$, a subsequent $X \otimes X$ parity measurement between ancilla and target, and a final Z basis readout of the ancilla qubit. In order to use this protocol for *logical* CNOTs, measurements of logical two-qubit parity operators are required, e.g., operators such as $Z_L \otimes Z_L$, which are nonlocal operators involving $2d$ physical qubits.

3.1 Nearest-neighbor CNOT

Let us first discuss standard lattice surgery between two neighboring wide qubits in the default encoding. Consider the CNOT protocol in Fig. 4. Lattice surgery [19] is a protocol for fault-tolerant logical parity measurements which only requires the measurement of local stabilizer operators. After initializing an ancilla qubit in the $|+\rangle$ state, lattice surgery between the Z edges of the control and ancilla qubit in step (2) measures their $Z_L \otimes Z_L$ parity. This is done by modifying the stabilizers along the boundaries. The boundary X stabilizers are merged to form four-qubit stabilizers (orange), and new Z stabilizers (blue) are introduced. While the stabilizers still mutually commute, this procedure increases the total number of stabilizers by one. In other words, the number of degrees of freedom is reduced by one, and one bit of information is measured during this proto-

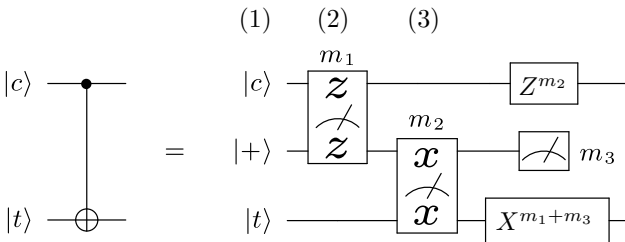


Figure 3: A CNOT between a control $|c\rangle$ and a target $|t\rangle$ is equivalent to a $Z \otimes Z$ parity measurement between $|c\rangle$ and an ancilla in the $|+\rangle$ state, followed by an $X \otimes X$ parity measurement between ancilla and $|t\rangle$, and finally a Z basis measurement of the ancilla. The measurement outcomes determine a Pauli correction.

col. The measurement outcome of the orange stabilizers is trivial, as they are products of previously known boundary stabilizers. The outcome of the blue stabilizers, on the other hand, is nontrivial. They contain each boundary qubit exactly once. Therefore, their product is precisely the operator $Z_{\text{edge}}^{(\text{control})} \otimes Z_{\text{edge}}^{(\text{ancilla})}$, which corresponds to the $Z_L \otimes Z_L$ parity in the default encoding. Thus, lattice surgery implements a fault-tolerant parity measurement between logical qubits. Similarly, in the following lattice surgery step (3), the blue stabilizers are trivial, and the product of orange stabilizers is

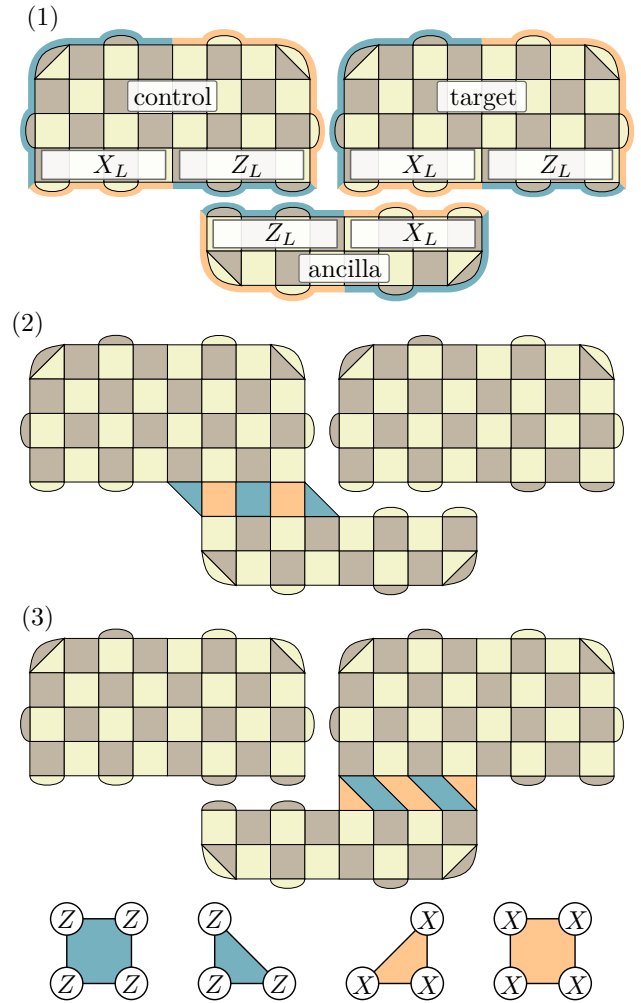


Figure 4: CNOT by lattice surgery corresponding to the gate circuit in Fig. 3. (1) All qubits are in the default encoding $X_{\text{edge}} = X_L$ and $Z_{\text{edge}} = Z_L$, and the ancilla is initialized in the $|+\rangle$ state. (2) To measure the $Z_L \otimes Z_L$ parity between control and target, the two-qubit boundary stabilizers are merged (orange), and new Z type stabilizers (blue) are introduced, whose product is precisely the parity. (3) Similarly, the $X_L \otimes X_L$ parity between ancilla and target is measured by the product of new X type stabilizers (orange).

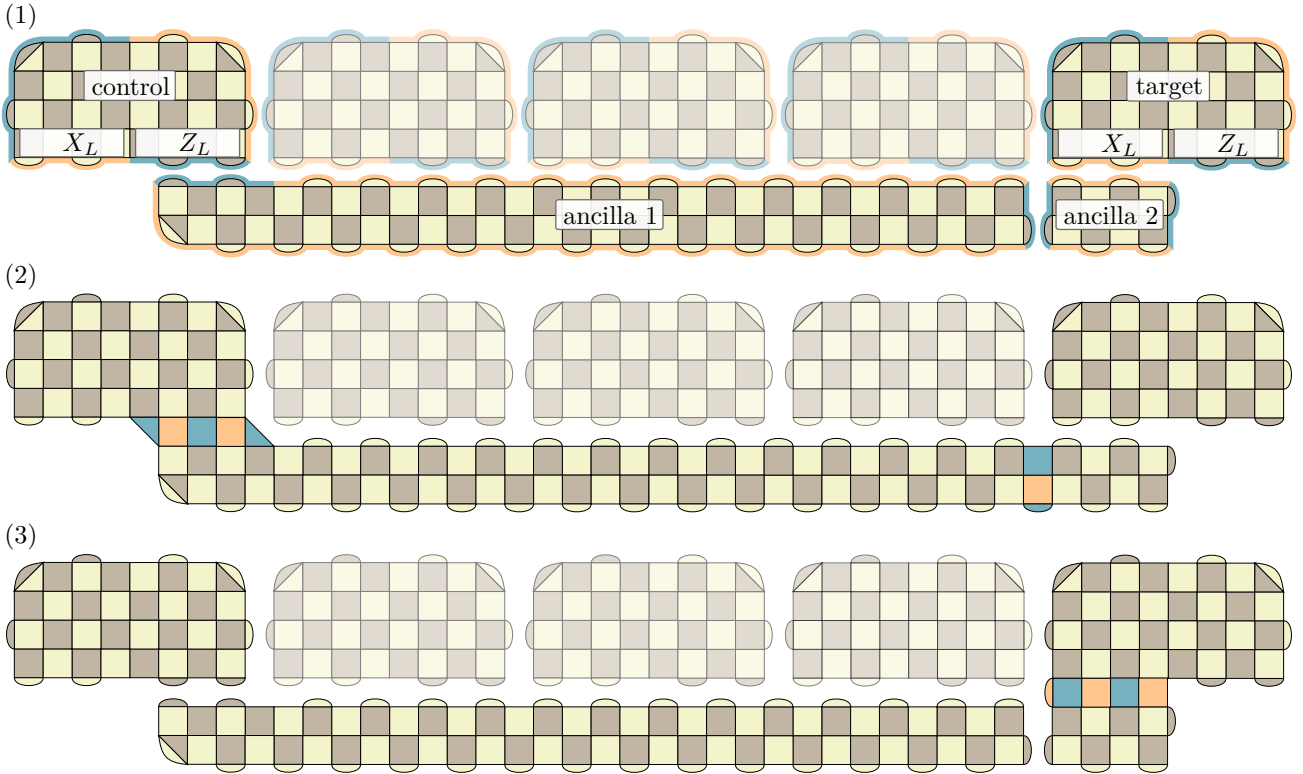


Figure 5: Long-range CNOT between two wide qubits in the default encoding that are separated by three other qubits. After initializing two ancillas in the $|+\rangle$ state (1), lattice surgery (2) simultaneously measures the $Z_L \otimes Z_L$ parities between control and ancilla 1, and ancilla 1 and ancilla 2. This also yields the $Z_L \otimes Z_L$ parity between control and ancilla 2, such that ancilla 2 can be used for an $X_L \otimes X_L$ parity measurement (3) with the target qubit. At the end of the CNOT protocol, ancilla 1 is read out in the X basis with outcome m , leading to a Z^m correction on the control.

$X_{\text{edge}}^{(\text{ancilla})} \otimes X_{\text{edge}}^{(\text{target})}$. A Z_L basis measurement of the ancilla qubit completes the gate circuit in Fig. 3. The subsequent Pauli corrections are Clifford gates and can be handled by edge tracking.

3.2 Long-range CNOT

A similar protocol can be used to perform CNOTs between logical qubits that are not nearest neighbors, but separated by some distance. For this, we use lattice surgery to measure the $Z_L \otimes Z_L$ parities between the control qubits and multiple ancilla qubits simultaneously [19, 22, 23]. In the protocol in Fig. 5, two ancilla qubits are initialized in the $|+\rangle$ state, one long ancilla that spans the entire distance between the control and target, and another that is adjacent to the X edge of the target. In step (2), lattice surgery simultaneously measures the $Z_L \otimes Z_L$ parities between control and long ancilla, and between both ancillas. This effectively measures the $Z_L \otimes Z_L$ parity between control and ancilla 2 as the product of both measurements. Thus, ancilla 2 can be used as the ancilla of the CNOT protocol of Fig. 3. An $X_L \otimes X_L$ parity measurement between an-

ancilla 2 and the target qubit, and a subsequent Z basis readout of ancilla 2 complete the CNOT protocol. Since ancilla 1 is still entangled with the control qubit, it cannot be discarded right away, but needs to be measured in the X basis with outcome $m \in \{0, 1\}$, which leads to a subsequent Z^m Pauli correction on the control qubit.

Vertical X error strings connecting the (orange) X edges of the long ancilla qubit can introduce errors to the CNOT protocol. While the number of possible error strings increases linearly with the control-target separation s , the probability of error strings decreases exponentially with the width of the ancilla. Therefore, the width needs to increase with $\mathcal{O}(\log s)$ in order to maintain the CNOT gate fidelity, implying a space overhead of $\mathcal{O}(s \log s)$ for the long-range CNOT. There are two factors that contribute to the time overhead of the protocol: decoding and syndrome readout errors. While decoding can be done with a runtime that scales with $\mathcal{O}(\log s)$ [24], the correction of stabilizer measurement errors is handled by recording multiple rounds of syndrome extraction for one code cycle [25]. This effectively introduces a third dimension to the code. The

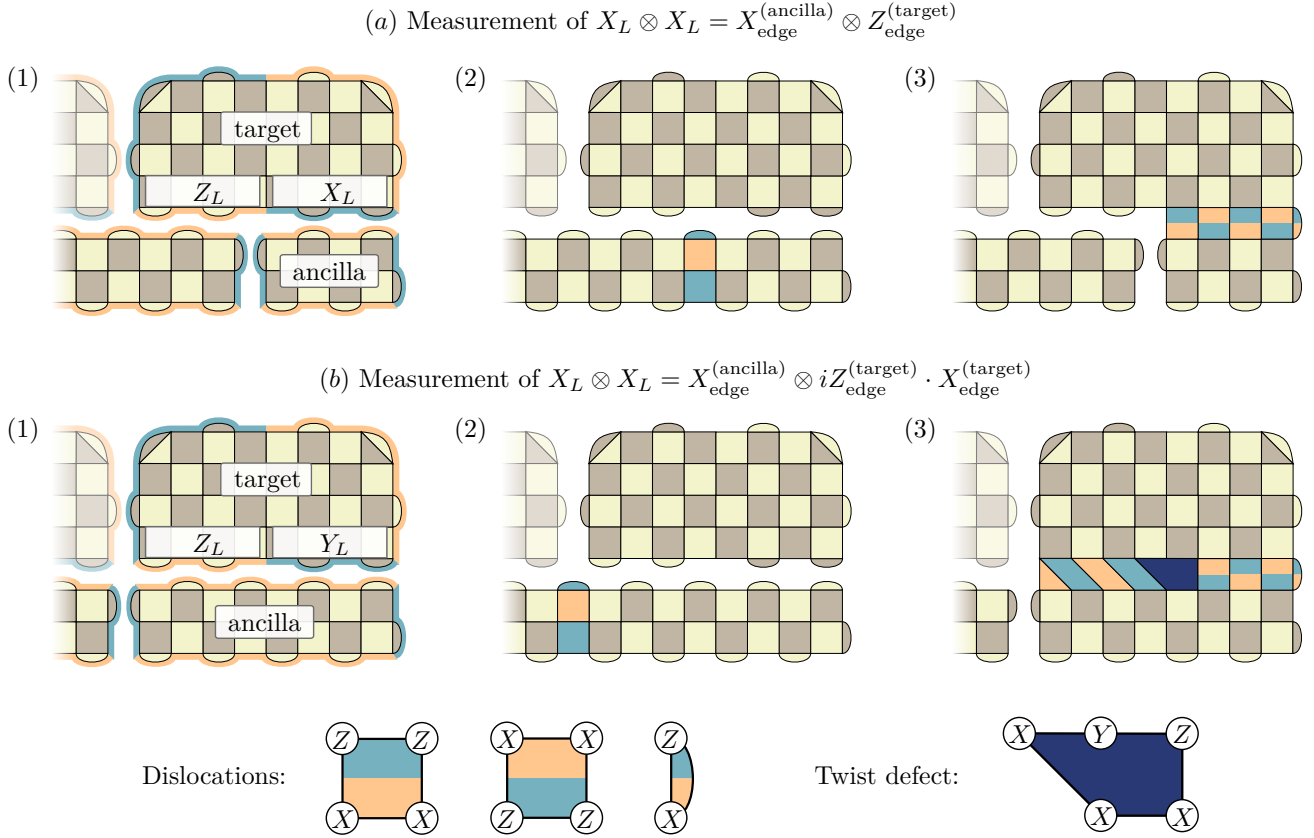


Figure 6: $X_L \otimes X_L$ parity measurements between an ancilla and an edge-tracked target qubit. In (a), edge tracking has changed the encoding of the target to $X_{\text{edge}} = Z_L$ and $Z_{\text{edge}} = X_L$. The stabilizer configuration that measures the $X_L \otimes X_L$ parity corresponds to a dislocation line. In (b), the encoding of the target qubit is $X_{\text{edge}} = Z_L$ and $Z_{\text{edge}} = Y_L$. Here, the $X_L \otimes X_L$ parity is measured by a stabilizer configuration that corresponds to a dislocation line that is terminated by a twist defect.

number of recorded measurement rounds for each code cycle depends on the measurement fidelity. With higher measurement fidelity, fewer measurement rounds are required to reach the same logical CNOT gate fidelity. As with the width of the long ancilla, errors in the time dimension are suppressed exponentially with the number of measurement rounds, i.e., with the code distance in time, but the number of possible error strings increases linearly with s . This implies that the number of measurement rounds needs to increase with $\mathcal{O}(\log s)$. Thus, the total time overhead is still just $\mathcal{O}(\log s)$, which is essentially constant for finite-size systems.

Note that in our figures (such as Fig. 5), the widths of the ancilla qubits, and therefore their code distances, are chosen to be smaller than the code distances of the wide qubits. This may be a valid choice for some computations, since the ancillas only need to survive for the duration of the CNOT, as opposed to data qubits that may need to survive for the entire computation. In practice, however, we expect that the space reserved for ancilla qubits will be in use for various CNOT gates for essentially the entire duration of the quantum compu-

ation. Therefore, for most applications, the code distances of the ancilla qubits and the data qubits should be chosen to be equal, and the logarithmic space overhead scaling with the control-target separation can be ignored. In this case, all logical qubits are protected against error strings of length $(d-1)/2$ during each code cycle. There is still a logarithmic space overhead scaling, since the necessary code distance to reach a certain target error probability at the end of a quantum computation involving n logical qubits scales with $\mathcal{O}(\log n)$.

3.3 CNOT between edge-tracked qubits

The previously discussed standard lattice surgery protocols can be used to measure $Z_{\text{edge}} \otimes Z_{\text{edge}}$ and $X_{\text{edge}} \otimes X_{\text{edge}}$. However, CNOTs between edge-tracked qubits may require additional parity measurements. This is where dislocations and twist defects come into play.

In Fig. 6, we explore the two additional situations that may occur for $X_L \otimes X_L$ parity measurements be-

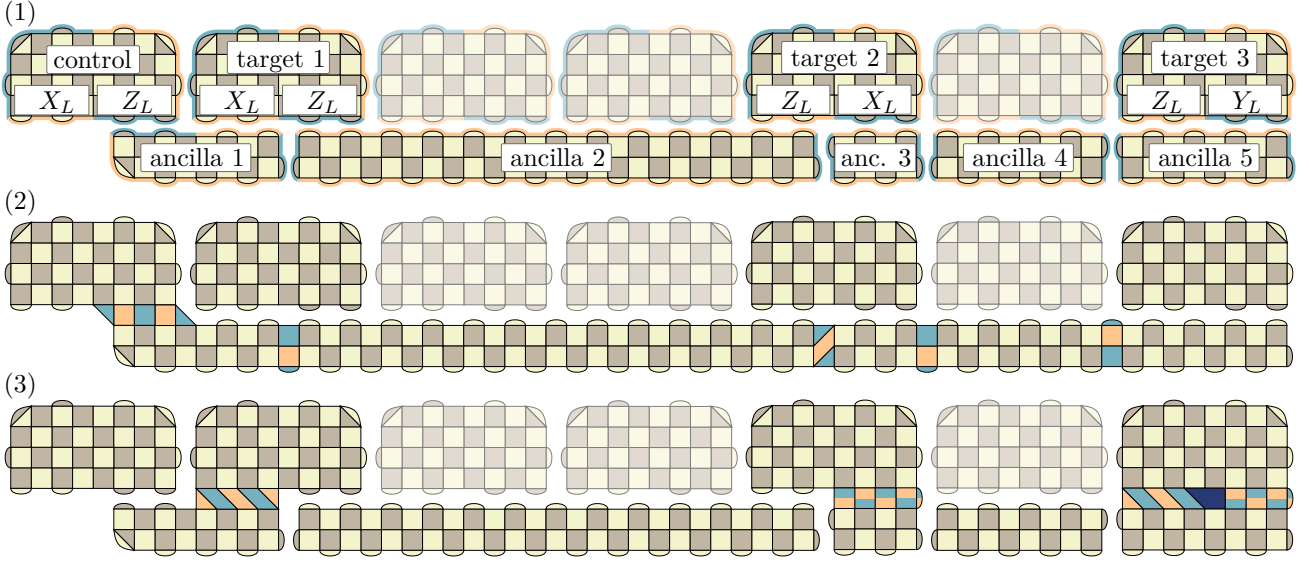


Figure 7: Long-range multi-target CNOTs with edge-tracked qubits. The control, ancillas, and target 1 are in the default encoding $Z_{\text{edge}} = Z_L$ and $X_{\text{edge}} = X_L$, whereas target 2 and target 3 have been modified by edge tracking, such that $X_L \otimes X_L$ parity measurements require lattice surgery between different edge types. Five ancilla qubits are initialized in the $|+\rangle$ state (1) and their $Z_L \otimes Z_L$ parities with the control qubit are measured simultaneously (2). Ancillas 2 and 4 merely provide long-range communication and are not used for CNOTs, but are instead read out in the X basis. Subsequent $X_L \otimes X_L$ parity measurements (3) use the previously discussed lattice surgery protocols for edge-tracked qubits.

tween an ancilla and an edge-tracked target qubit during a CNOT protocol. In the first situation (a), the X_L operator is defined by the target's Z edge as a consequence of edge tracking. Thus, lattice surgery needs to measure the operator $X_{\text{edge}}^{(\text{ancilla})} \otimes Z_{\text{edge}}^{(\text{target})}$. For this, the boundary stabilizers are merged, and new stabilizers are introduced. One can check that all stabilizers commute, and that the product of the nontrivial stabilizers indeed yields $X_L \otimes X_L$.

The remaining possibility is that, as a consequence of edge tracking, none of the edges of the target define its X_L . In (b), the target qubit is in the encoding where $X_{\text{edge}} = Z_L$ and $Z_{\text{edge}} = Y_L$. Since $X_L = iZ_L Y_L$, and therefore $X_L = iX_{\text{edge}} Z_{\text{edge}}$, lattice surgery now needs to measure $X_{\text{edge}}^{(\text{ancilla})} \otimes iX_{\text{edge}}^{(\text{target})} \cdot Z_{\text{edge}}^{(\text{target})}$. Similar to the previous cases, stabilizers along the boundary in (b3) are merged yielding the trivial stabilizers. The product of the newly introduced nontrivial stabilizers is again the $X_L \otimes X_L$ parity. Note that the center qubit of the blue five-qubit operator contributes to the stabilizer measurement in the Y basis, since it is part of both the X and the Z edge.

The three different lattice surgeries in panel (3) of Fig. 5, and panels (a3) and (b3) of Fig. 6 can also be interpreted as protocols to measure $X_L \otimes X_L$, $Z_L \otimes X_L$ and $Y_L \otimes X_L$ between a wide qubit and a square qubit in the default encoding. The protocol involving Y_L is what we refer to as twist-based lattice surgery, since the

five-qubit operator corresponds to a twist defect.

Such a parity measurement can also be used to measure the product $iX_{\text{edge}} \cdot Z_{\text{edge}}$ of a qubit, e.g., to read out the qubit in the Y_L basis in the default encoding. For this, an ancilla can be initialized in the $|0\rangle$ state, such that a $Y_L^{(\text{qubit})} \otimes Z_L^{(\text{ancilla})}$ parity measurement between qubit and ancilla is equivalent to a Y_L measurement of the qubit.

This covers all the necessary lattice surgery protocols for CNOTs between edge-tracked qubits. The $Z_L \otimes Z_L$ parity measurements between ancilla qubits and edge-tracked control qubits are analogous to the $X_L \otimes X_L$ parity measurements in Fig. 6. The concrete implementation of the required stabilizer measurements depends on the given architecture. While Majorana-based implementations allow for direct measurements of the necessary operators, non-topological setups such as superconducting qubits require the use of measurement qubits. In the latter case, the stabilizer measurement protocol requires special care in order to avoid correlated errors that lower the effective code distance, as we show in Appendix A.

3.4 Connection to twist defects

The stabilizer configurations in these modified lattice surgery protocols feature dislocations and twist defects. The mixed stabilizers in (a3) correspond to a dislocation in the surface code. The stabilizer configuration in (b3)

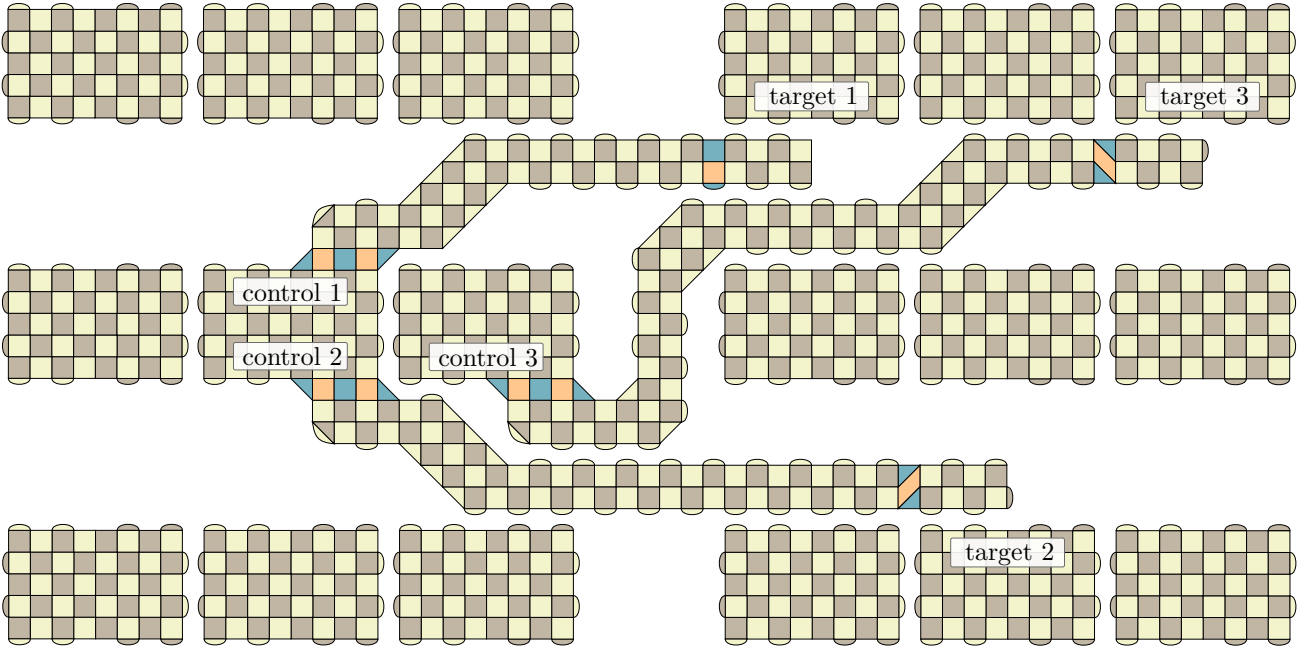


Figure 8: Example of a two-dimensional arrangement of surface code qubits, where qubits are grouped in blocks of six. The long ancilla qubits can be used for three simultaneous long-range CNOT gates.

corresponds to a dislocation line between the X edge of the ancilla and the Z edge of the target which is terminated by a five-qubit twist defect [15, 16].

Twist-based lattice surgery can also be interpreted in a Majorana fermion picture. It was pointed out that the corners of square surface code qubits (as in Fig. 1) correspond to twist defects [16]. Similarly, the ends of the X and Z edges of wide qubits can be replaced by twist defects – i.e., Majorana fermions – such that the logical operators X_L , Z_L , and Y_L are two-Majorana fermion parity operators. Lattice surgery then effectively implements a four-Majorana fermion parity measurement [16]. In Fig. 6 (b3), these four Majorana fermions are in the bottom left and right corners of the target, and in the top left and right corners of the ancilla. The twist defect corresponds to the remaining Majorana fermion residing between the X and Z edge of the target qubit, which is not part of the parity measurement.

3.5 Long-range multi-target CNOT

The simultaneous $Z_L \otimes Z_L$ parity measurements of long-range CNOTs can be used for multi-target CNOTs, i.e., for multiple CNOTs with the same control, but different target qubits. An example is shown in Fig. 7, where five ancillas are used to perform three CNOTs with three edge-tracked targets simultaneously. Step (2) shows the simultaneous measurement of $Z_L \otimes Z_L$ parities of six

neighboring qubits, which correspond to one control and five ancilla qubit. This protocol effectively measures the $Z_L \otimes Z_L$ parities of all pairs of qubits, and in particular of the control and each ancilla qubit. Thus, each of the five ancilla qubits can be used for $X_L \otimes X_L$ parity measurements with target qubits. While ancillas 1, 3, and 5 are used for CNOTs with targets 1, 2, and 3, ancillas 2 and 4 merely bridge distances between distant qubits.

Thus, by simultaneously initializing multiple ancillas, lattice surgery provides long-range multi-target CNOTs with edge-tracked qubits with the same time overhead as single CNOTs. At the end of the protocol, ancillas that are used for $X_L \otimes X_L$ parity measurements with target qubits are read out in the Z basis, whereas ancillas used to bridge long distances are read out in the

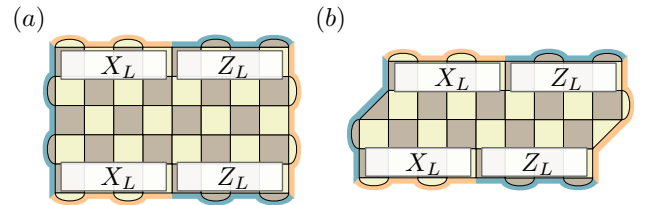


Figure 9: Double-sided qubits encode two logical qubits using (a) $2d^2 + d - 1$ or (b) $2d^2 - d$ physical qubits. The left and right edges correspond to the logical operators $Z_L \otimes Z_L$ and $X_L \otimes X_L$ of both encoded qubits, respectively.

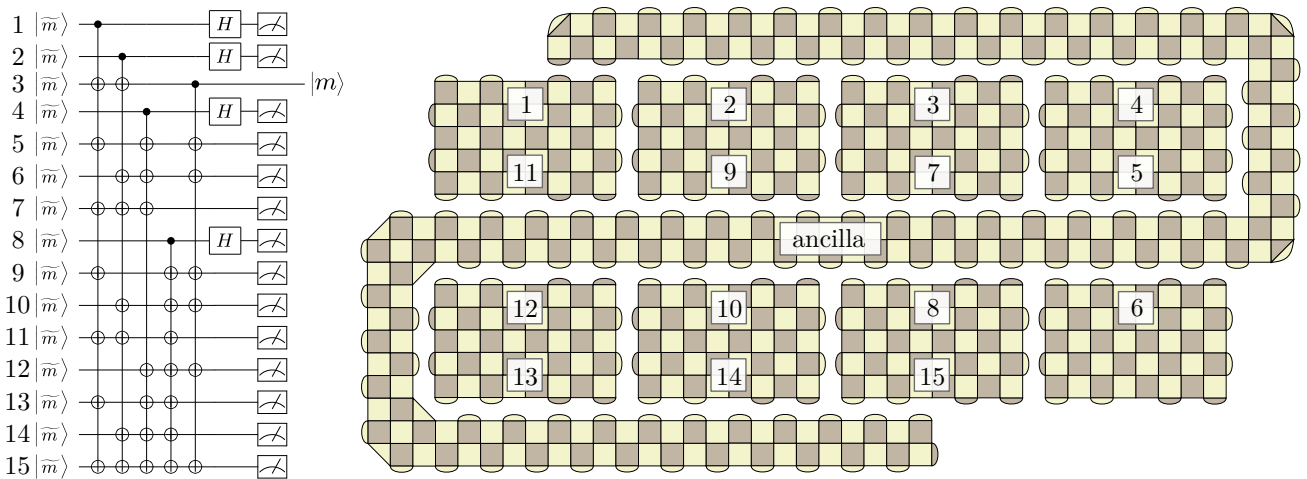


Figure 10: Example of the 15-to-1 magic state distillation protocol using long-range multi-target CNOTs via lattice surgery. By appropriately partitioning the long ancilla qubit, it can be used for each of the five multi-target CNOTs of the protocol.

X basis. Multi-target CNOTs are particularly useful for logical T gates, as magic state distillation schemes typically consist of many multi-target CNOTs. These complete the universal gate set of our scheme, as we discuss in the following section.

4 2D arrangement of logical qubits

So far, we have considered logical qubits arranged on a line. The lattice-surgery-based CNOT gates can also provide long-range connectivity in two dimensions. For this, it will be convenient to use the space of wide qubits to encode *two* logical qubits instead of just one. The double-sided qubits shown in Fig. 9 reduce the space overhead from $\sim 2d^2$ physical qubits for each logical (wide) qubit back to $\sim d^2$ physical qubits, similar to the square qubits in Fig. 1. The downside of double-sided qubits is that state initialization and readout is more complicated, as the two encoded qubits cannot be measured independently. However, one can use lattice surgery to initialize and read out in any Pauli basis. For instance, a qubit can be initialized in the $|0\rangle$ state by initializing a standard ancilla encoding a single qubit in the $|0\rangle$ state and performing lattice surgery via the Z edges of both qubits. Readout is done the same way, using the appropriate edge of the qubit. Should one require fast initialization and readout, it is still possible to use wide qubits instead of double-sided qubits.

An example of a 2D arrangement of double-sided qubits is shown in Fig. 8, where they form blocks of six logical qubits. The space between blocks is used for ancilla qubits for long-range CNOT gates. The separation between blocks not only sets the maximum width of the ancilla qubits, but also influences the number

of multi-target CNOTs that can be performed simultaneously. The larger the separation, the more ancilla qubits can fit between the qubit blocks. The example in Fig. 8 shows three simultaneous CNOT gates, where the space between qubit blocks allows for two parallel “lanes” of CNOT ancillas. Thus, a larger separation between qubit blocks increases the connectivity, but also the space overhead.

4.1 Example: Magic state distillation

Having discussed the implementation of the logical Clifford gates in our scheme, the remaining gate for universal quantum computation is the logical T gate. One possibility to implement the logical T gate using physical T gates and logical Clifford gates is magic state distillation [20]. The aim of this scheme is to generate an encoded magic state $|m\rangle = (|0\rangle + e^{i\pi/4}|1\rangle)/\sqrt{2}$, which corresponds to a $|+\rangle$ -state on which a T gate has been performed. A CNOT gate between $|m\rangle$ and a target qubit, followed by the measurement of $|m\rangle$ corresponds to a logical T gate on the target qubit, up to a Clifford correction.

However, it is only possible to prepare *physical* magic states, which are moreover faulty states $|\tilde{m}\rangle$, i.e., generated using an imprecise physical T gate. These physical states can be converted into *logical* faulty magic states $|\tilde{m}\rangle$ via state injection [19]. Magic state distillation protocols take many faulty magic states and convert them to fewer, but more precise magic states. These protocols typically consist of many multi-target CNOT gates.

One example of a magic state distillation protocol is shown in Fig. 10 for the example of 15-to-1 conversion [20], which converts 15 faulty magic states into one better magic state. It consists of 34 CNOT gates

	color code	wide surface code	double-sided surface code
space overhead	+ low ($\approx \frac{3}{4}d^2$ or $\frac{1}{2}d^2$)	- high ($\approx 2d^2$)	\sim moderate ($\approx d^2$)
initialization & readout	+ fast X, Y, Z	\sim fast X, Z ; slow Y	- slow X, Y, Z
stabilizer weight	- high (six or eight)	+ low (four)	+ low (four)

Table 1: Comparison between color-code-based [13, 23] and surface-code-based schemes.

grouped into five multi-target CNOTs. The figure also shows an arrangement of qubits that can be used to implement the protocol. By appropriately partitioning the long ancilla qubit, each of the five multi-target CNOTs can be performed using the protocol in Fig. 7. We provide the detailed stabilizer configurations for this 15-to-1 conversion in Appendix B. The space overhead of the 15-to-1 conversion depends on the code distances of the magic states and the width of the ancilla. The time overhead is mostly determined by the five multi-target CNOTs, which require two code cycles (including repetitions accounting for stabilizer measurement errors) for their parity measurements by lattice surgery.

5 Conclusion

We have demonstrated that edge tracking can be used to eliminate the time overhead of logical single-qubit Clifford gates in surface codes, as should be expected considering the Gottesman-Knill theorem. Twist-based lattice surgery provides long-range multi-target CNOTs with a time overhead that only scales with $\mathcal{O}(\log s)$ of the control-target separation s , and a space overhead that scales with $\mathcal{O}(s \log s)$. Compared to color code qubits, the surface code qubits used in our scheme require more physical qubits ($\sim d^2$) for each logical qubit with code distance d , but – with the exception of twist defects – only require the measurement of weight-four stabilizers. Our scheme can provide full 2D connectivity between logical qubits, where the degree of connectivity is governed by the separation of qubit blocks, and therefore by the space overhead. Together with magic state distillation, our scheme allows for fault-tolerant universal quantum computation.

One may be wondering whether there is still any ad-

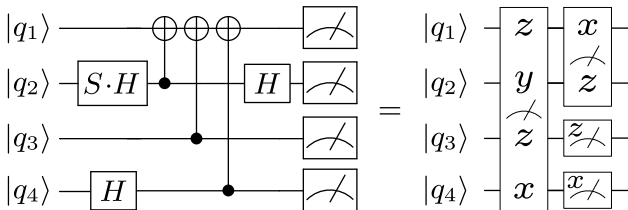


Figure 11: Example of a Clifford circuit that is reduced to Pauli product measurements.

vantage offered by the transversal single-qubit Clifford gates of color codes and the color-code-based lattice-surgery scheme presented in Ref. [23]. A comparison of these codes is shown in Tab. 1. While color codes require the measurement of higher-weight stabilizers, they offer fast qubit readout in all Pauli bases, and a lower space overhead of $\sim \frac{3}{4}d^2$ physical qubits per logical qubit for 6.6.6 color codes, or even $\sim \frac{1}{2}d^2$ for 4.8.8 color codes. So if the measurement of higher-weight stabilizers is not substantially more difficult in a given physical implementation, as might be the case for Majorana-based qubits, it is advantageous to use the color-code-based scheme. In other implementations, such as superconducting qubits, the difficulty of higher-weight stabilizer measurements shifts the preference towards surface-code-based architectures.

An important point is that the Gottesman-Knill theorem allows for the classical tracking of all Clifford gates, including CNOT gates. As CNOT gates map $X \otimes \mathbb{1} \rightarrow X \otimes X$ and $\mathbb{1} \otimes Z \rightarrow Z \otimes Z$, tracking of CNOTs does not preserve the locality of the logical operators, in

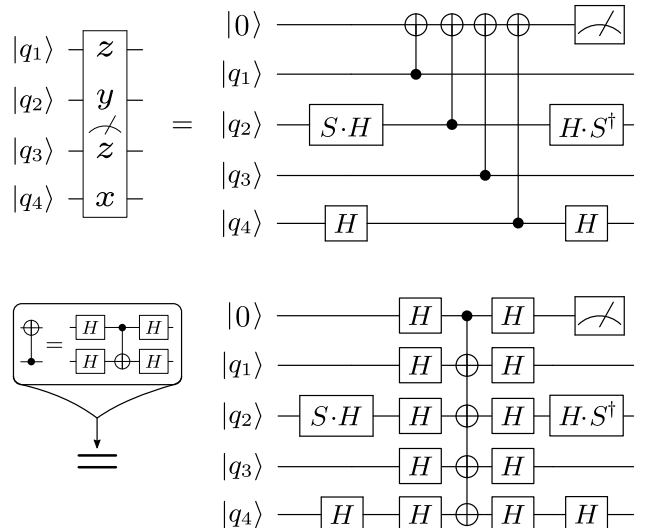


Figure 12: Circuit identity for the measurement of the Pauli product operator $Z \otimes Y \otimes Z \otimes X$ using an ancilla and a multi-target CNOT gate. The circuit identity exploits the fact that the roles of control and target can be reversed by the application of Hadamard gates before and after a CNOT gate. Any product of Pauli operators can be measured this way.

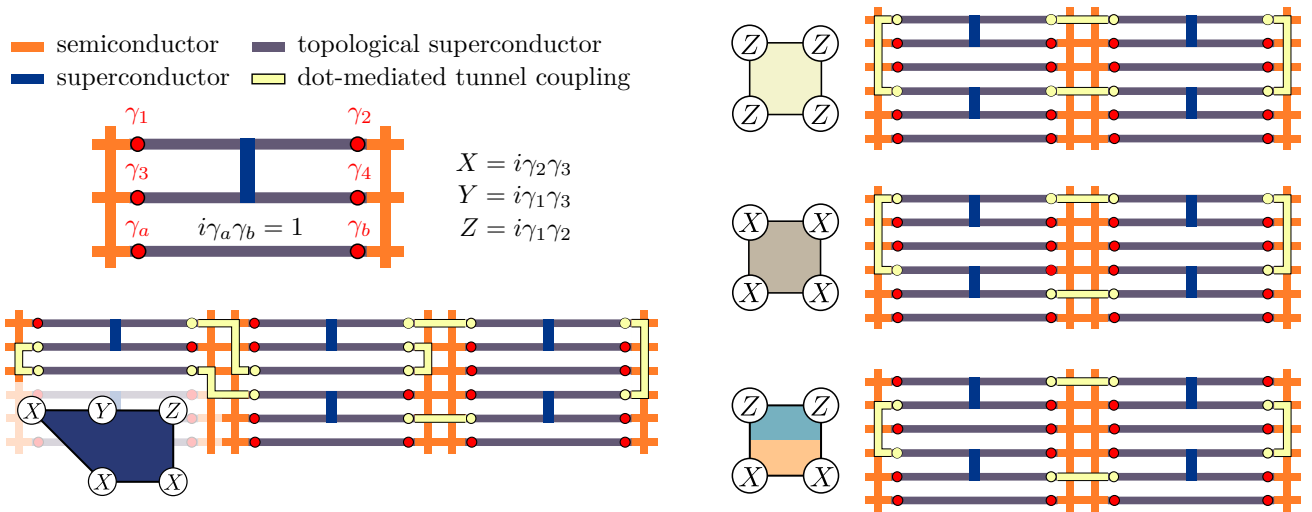


Figure 13: Tunnel coupling configurations for the measurement of various check operators using a square network of Majorana-based tetron qubits, as introduced in Ref. [21].

contrast to single-qubit Clifford gates. By tracking *all* Clifford gates, any layer of Clifford gates followed by n single-qubit measurements can always be compressed to n measurements of *nonlocal* products of Pauli operators without any preceding gate operations (see Fig. 11 for an example). With distilled magic states as a resource, any non-Clifford gate corresponds to trackable Clifford gates and a measurement of the magic state. In this case, Pauli product measurements are the only hardware operations that need to be performed explicitly. The fault-tolerant measurement of any nonlocal Pauli product can be implemented using an ancilla qubit and a multi-target CNOT gate on edge-tracked qubits. An example of such a protocol is shown in Fig. 12 for the measurement of the Pauli product $Z_L \otimes Y_L \otimes Z_L \otimes X_L$. Thus, any quantum computation can be performed using only two types of hardware operations: distillation of resource states and Pauli product measurements via multi-target CNOT gates on edge-tracked qubits.

A crucial problem of quantum information theory is the optimization of quantum circuits in order to minimize the space-time overhead of any quantum computation. However, any circuit optimization depends on the constraints set by the quantum computer hardware and the code used for error correction. Based on the existing schemes for surface-code and color-code quantum computation, the following minimal assumptions concerning the underlying hardware and the logical operations accessible by the code appear reasonable: (i) The underlying hardware can measure local products of physical Pauli operators. (ii) The quantum error-correcting code allows for the measurement of nonlocal products of logical Pauli operators. (iii) Resource states can be

generated for the implementation of logical non-Clifford gates. Based on these constraints, an important circuit optimization problem is to find heuristics that minimize the number of required resource states and the number of layers of Pauli product measurements, as these are the only operations that cannot be relegated to a classical computer.

Open questions related to our surface-code scheme include the efficient decoding of wide, long and double-sided qubits, estimations of their survival times, and implementations of our scheme in a concrete physical system. Our scheme may also be adapted to surface-code quantum computing with twist-based triangle codes [26], in order to avoid the reorientation of triangles, and to further reduce the space overhead of surface codes. We hope that our lattice-surgery-based approach can contribute to ongoing efforts to realize a surface-code quantum computer.

Acknowledgments

We thank Benjamin J. Brown, Jens Eisert, Markus S. Kesselring, and Fernando Pastawski for insightful discussions. This work has been supported by the Deutsche Forschungsgemeinschaft (Bonn) within the network CRC TR 183.

A Stabilizer measurements in concrete implementations

Our twist-based surgery scheme requires the measurement of certain operators that are products of Pauli op-

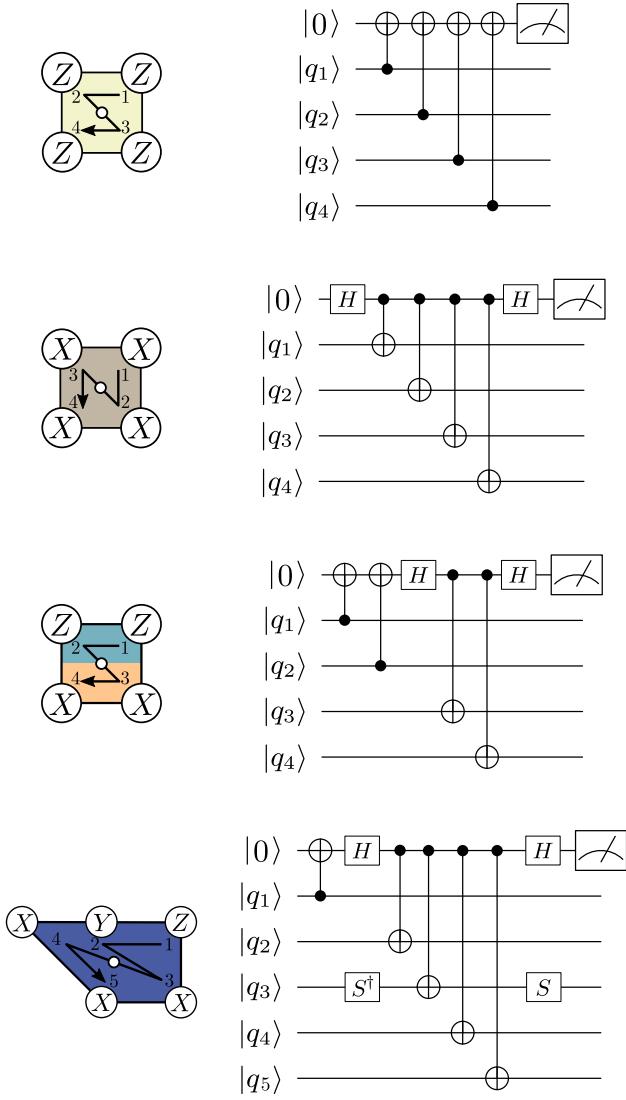


Figure 14: Circuits for the stabilizer readout using an ancillary measurement qubit placed in the center of the stabilizer.

erators on up to 5 qubits. How these operators are measured in practice depends on the concrete hardware used for quantum computing. In this appendix, we show how these measurements could in principle be implemented with Majorana-based qubits, and with non-topological qubits such as superconducting qubits that require the use of ancillary measurement qubits for stabilizer readout.

A.1 Majorana-based qubits

The primary operation of Majorana-based qubits is the measurement of local products of Majorana operators, which correspond to local products of Pauli operators. Thus, they can be straightforwardly used to measure

the stabilizers in our twist-based surgery scheme. In Fig. 13, we show how this can be done in a network of tetron qubits introduced in Ref. [21]. In a nutshell, these are qubits that are encoded in the doubly degenerate ground-state space of four Majorana zero modes $\gamma_1 \dots \gamma_4$ that are localized at the ends of two topological superconducting nanowires which are put into a fixed parity sector ($\gamma_1 \gamma_2 \gamma_3 \gamma_4 = -1$) by a non-topological superconductor bridging the two wires. The Majorana operators are self-conjugate $\gamma = \gamma^\dagger$ and mutually anti-commute $\{\gamma_i, \gamma_j\} = 2\delta_{i,j}$. Therefore, the Pauli operators of each tetron qubit can be chosen as $Z = i\gamma_1 \gamma_2$ and $X = i\gamma_2 \gamma_3$.

In a square lattice of tetrons, each tetron qubit is connected to a network of semiconductors. Local products of Majorana operators are measured by opening tunnel couplings between tetrons and the semiconductor network, such that the tunnel couplings form closed paths. The semiconducting wire segments between tetrons form quantum dots whose energy levels are shifted by virtual processes that tunnel electrons around the closed path. Since these processes involve each Majorana operator along the path exactly once, spectroscopy on any of the dots can be used to measure the product of the Majorana operators along the path. In Fig. 13, we show tunnel coupling configurations that can be used to measure X and Z stabilizers, dislocation operators, and twist operators. For the twist operator, in particular, additional Majoranas γ_a and γ_b in a fixed parity sector $i\gamma_a \gamma_b = 1$ are used as so-called coherent links in order to form the closed path. More details on operator measurements in tetron networks are found in Refs. [21, 23].

A.2 Non-topological qubits

For non-topological qubits such as superconducting qubits, Pauli products cannot be measured directly, but are usually read out using ancilla qubits (measurements qubits) that are located in the center of each stabilizer operator, such as in the scheme of Ref. [14]. These measurement qubits are entangled via two-qubit gates with each data qubit that is part of the stabilizer. Afterwards, they are read out to yield the corresponding Pauli product. The readout can be done using the circuits shown in Fig. 14. Depending on the elementary operations accessible in a given hardware, a different (but equivalent) circuit may be used, but in any case the readout of each stabilizer requires up to 5 two-qubit gates which need to be performed in succession.

One practical problem of this approach to stabilizer measurements is that due to the use of two-qubit gates, single errors on measurement qubits can spread to multiple data qubits. This can lead to correlated errors

which are referred to as hook errors [25]. In particular, Z errors on measurement qubits of Z stabilizers can lead to correlated Z errors on the surrounding data qubits. Similarly, X errors on X measurement qubits lead to correlated X errors. Since three errors are equivalent to just one error (up to a multiplication with a stabilizer), the worst case is the case of one error on a measurement qubit leading to two errors on data qubits. Since these errors will occur on the first two (or last two) qubits that were part of entangling two-qubit gates, the order of the two-qubit gates in the circuits of Fig. 14 is important.

The aim is to avoid these correlated errors from lowering the effective code distance. That is, we need to find an ordering of the two-qubit gates, such that a logical operator of a distance d qubit can only be formed by no fewer than d errors. For square surface code patches (as in Fig. 1), this is done by orienting the ordering of CNOT gates for X stabilizers in an N shape (or in a \mathcal{N} shape), and for Z stabilizers in a Z shape (or \mathcal{Z} shape), as was shown in Ref. [27]. This guarantees that correlated Z errors only form in the horizontal direction, while logical Z strings are all oriented vertically. Similarly, correlated X errors form horizontally, which does not contribute towards a vertical logical X string.

However, in our scheme, we use the double-sided qubits of Fig. 9, which have Z and X operators in both the horizontal and vertical direction. Thus, it is not sufficient to assign one orientation to the Z stabilizers and the other to X stabilizers. In fact, the left half of the double-sided qubit looks like the square qubit in Fig. 1, i.e., Z stabilizers should be oriented in a Z shape, and X stabilizers in an N shape. In contrast, the right half of the double-sided qubit looks like a rotated square qubit, i.e., Z stabilizer should be oriented in an N shape, and X stabilizers in a Z shape. In the crossover region in the center, both logical X and Z operators are vertical strings, such that both Z and X stabilizers should be oriented in a Z shape. This motivates the first condition for a valid ordering shown in Fig. 15a. Z stabilizers in the blue region should be oriented in a Z shape, and in an N shape outside of the blue region. X stabilizers in the red region should be oriented in a Z shape, and in an N shape outside of the red region. One can verify that with this choice of orientations, no logical operator string can be formed with fewer than d physical errors.

The only remaining problem is the scheduling of the two-qubit gates. Since the largest check operator is the 5-qubit twist defect, each two-qubit gate needs to be assigned to one of 5 time steps. This implies two other conditions on the ordering of the two-qubit gates. Since each data qubit can only be part of one two-qubit gate in a given time step, the four time steps assigned to the two-qubit gates that a given data qubit is part of

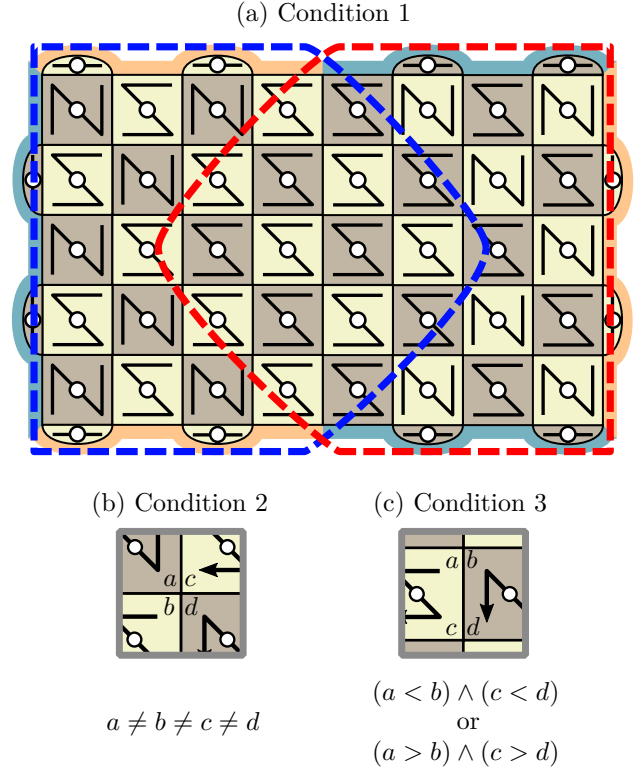


Figure 15: Three conditions for a valid ordering of two-qubit gates during stabilizer readout. (a) In the blue region, the Z stabilizers are oriented in a Z shape, and outside of this region in an N shape. In the red region, the X stabilizers are oriented in a Z shape, and outside of this region in an N shape. This ensures that hook errors do not lower the effective code distance. (b) The time steps a , b , c , and d , that are assigned to the (up to) four two-qubit gates of one data qubits need to be all different. (c) For all edges between neighboring stabilizers, the condition shown in the figure needs to be fulfilled to ensure that the sequence of two-qubit measurements reproduces the desired stabilizer measurements.

need to be all different, which is the second condition in Fig. 15b.

Finally, one needs to ensure that the sequence of two-qubit measurements reproduces the desired stabilizer measurements. For this, consider the action of the CNOT gates of the readout circuit for a Z stabilizer in the Heisenberg picture (or see Appendix B of Ref. [14]). The aim of the readout circuit is to map the Z operator of the measurement qubit onto the operator $Z^{\otimes 5}$ on the measurement qubit and all four data qubits. Since a CNOT maps $\mathbb{1} \otimes Z$ onto $Z \otimes Z$ (where the first qubit is the control and the second is the target), the circuit in Fig. 14 achieves exactly that. However, notice that two CNOT gates of the readout circuit of a neighboring X stabilizer also act on the Z operators of the data qubits. These may map the operator onto an operator

that involves the Z operator of the wrong measurement qubit, i.e., the measurement qubit of the neighboring stabilizer. This needs to be avoided, as this neighboring measurement qubit is measured in X , which anti-commutes with the aforementioned operator, leading to random measurement outcomes. For concreteness, let us refer to the time steps assigned to the two CNOTs of the Z stabilizer in question as a and c , and to the CNOTs of the X stabilizer as b and d , as in Fig. 15c. There are only two choices of time steps that map the Z operators of the measurement qubit to the correct $Z^{\otimes 5}$ operator. The first possibility is that $a > b$ and $c > d$, such that the two CNOTs of the neighboring X stabilizer have already been performed, which precludes them from acting on the Z operator of the data qubit. The second possibility is that $a < b$ and $c < d$, such that the mapping is performed twice: The first CNOT maps $\mathbb{1} \otimes Z$ to $Z \otimes Z$, and the second CNOT maps $Z \otimes Z$ back to $\mathbb{1} \otimes Z$. This is the third condition shown in Fig. 15c. It needs to hold for all edges between neighboring stabilizers.

It is possible to find an ordering of two-qubit gates in 5 time steps (due to the 5-qubit twist operators) that fulfills all three conditions. Such a possibility is shown in Fig. 16. The figure shows the most generic situation which involves the bulk stabilizers of standard and double-sided qubits, as well as a dislocation line and a twist defect.

B Magic state distillation protocol

Here, we explicitly show the lattice surgery protocols for the multi-target CNOTs part of the 15-to-1 magic state distillation scheme in Fig. 10. Figures 17 and 18 show the five multi-target CNOTs of the distillation protocol, where the control and target qubits are highlighted in blue and orange, respectively. Note that the default encodings of the X and Z edges of qubits 5, 9 and 11 are inverted in this protocol. The figures only show the $Z_L \otimes Z_L$ parity measurements. The subsequent $X_L \otimes X_L$ parity measurements are done via lattices surgeries between the highlighted orange edges and the adjacent ancilla qubits.

References

- [1] J. Preskill, *Reliable quantum computers*, *Proc. Roy. Soc. Lond. A* **454**, 385 (1998).
- [2] A. Kitaev, *Fault-tolerant quantum computation by anyons*, *Ann. Phys.* **303**, 2 (2003).
- [3] B. M. Terhal, *Quantum error correction for quantum memories*, *Rev. Mod. Phys.* **87**, 307 (2015).
- [4] M. H. Devoret and R. J. Schoelkopf, *Superconducting circuits for quantum information: An outlook*, *Science* **339**, 1169 (2013).
- [5] D. Loss and D. P. DiVincenzo, *Quantum computation with quantum dots*, *Phys. Rev. A* **57**, 120 (1998).
- [6] R. M. Lutchyn, E. P. A. M. Bakkers, L. P. Kouwenhoven, P. Krogstrup, C. M. Marcus, and Y. Oreg, *Realizing Majorana zero modes in superconductor-semiconductor heterostructures*, [arXiv:1707.04899](https://arxiv.org/abs/1707.04899) (2017).
- [7] D. Gottesman, *Stabilizer codes and quantum error correction*, *Ph.D. thesis*, California Institute of Technology (1997).
- [8] S. B. Bravyi and A. Y. Kitaev, *Quantum codes on a lattice with boundary*, [arXiv:quant-ph/9811052](https://arxiv.org/abs/quant-ph/9811052) (1998).
- [9] E. T. Campbell, B. M. Terhal, and C. Vuillot, *Roads towards fault-tolerant universal quantum computation*, *Nature* **549**, 172 (2017).
- [10] D. S. Wang, A. G. Fowler, A. M. Stephens, and L. C. L. Hollenberg, *Threshold error rates for the toric and planar codes*, *Quantum Info. Comput.* **10**, 456 (2010).
- [11] R. S. Andrist, H. G. Katzgraber, H. Bombin, and M. A. Martin-Delgado, *Error tolerance of topological codes with independent bit-flip and measurement errors*, *Phys. Rev. A* **94**, 012318 (2016).
- [12] H. Bombin and M. A. Martin-Delgado, *Topological quantum distillation*, *Phys. Rev. Lett.* **97**, 180501 (2006).
- [13] A. J. Landahl, J. T. Anderson, and P. R. Rice, *Fault-tolerant quantum computing with color codes*, [arXiv:1108.5738](https://arxiv.org/abs/1108.5738) (2011).
- [14] 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).
- [15] H. Bombin, *Topological order with a twist: Ising anyons from an abelian model*, *Phys. Rev. Lett.* **105**, 030403 (2010).
- [16] B. J. Brown, K. Laubscher, M. S. Kesselring, and J. R. Wootton, *Poking holes and cutting corners to achieve Clifford gates with the surface code*, *Phys. Rev. X* **7**, 021029 (2017).
- [17] M. B. Hastings and A. Geller, *Reduced space-time and time costs using dislocation codes and arbitrary ancillas*, *Quantum Info. Comput.* **15**, 962 (2015).
- [18] D. Gottesman, *The Heisenberg representation of quantum computers*, *Proc. XXII Int. Coll. Group. Th. Meth. Phys.* **1**, 32 (1999).
- [19] C. Horsman, A. G. Fowler, S. Devitt, and R. V. Meter, *Surface code quantum computing by lattice surgery*, *New J. Phys.* **14**, 123011 (2012).

- [20] S. Bravyi and A. Kitaev, *Universal quantum computation with ideal Clifford gates and noisy ancillas*, *Phys. Rev. A* **71**, 022316 (2005).
- [21] T. Karzig, C. Knapp, R. M. Lutchyn, P. Bonderson, M. B. Hastings, C. Nayak, J. Alicea, K. Flensberg, S. Plugge, Y. Oreg, C. M. Marcus, and M. H. Freedman, *Scalable designs for quasiparticle-poisoning-protected topological quantum computation with Majorana zero modes*, *Phys. Rev. B* **95**, 235305 (2017).
- [22] D. Litinski, M. S. Kesselring, J. Eisert, and F. von Oppen, *Combining topological hardware and topological software: Color-code quantum computing with topological superconductor networks*, *Phys. Rev. X* **7**, 031048 (2017).
- [23] D. Litinski and F. von Oppen, *Braiding by Majorana tracking and long-range CNOT gates with color codes*, *Phys. Rev. B* **96**, 205413 (2017).
- [24] G. Duclos-Cianci and D. Poulin, *Fast decoders for topological quantum codes*, *Phys. Rev. Lett.* **104**, 050504 (2010).
- [25] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, *Topological quantum memory*, *Journal of Mathematical Physics* **43**, 4452 (2002).
- [26] T. J. Yoder and I. H. Kim, *The surface code with a twist*, *Quantum* **1**, 2 (2017).
- [27] Y. Tomita and K. M. Svore, *Low-distance surface codes under realistic quantum noise*, *Phys. Rev. A* **90**, 062320 (2014).

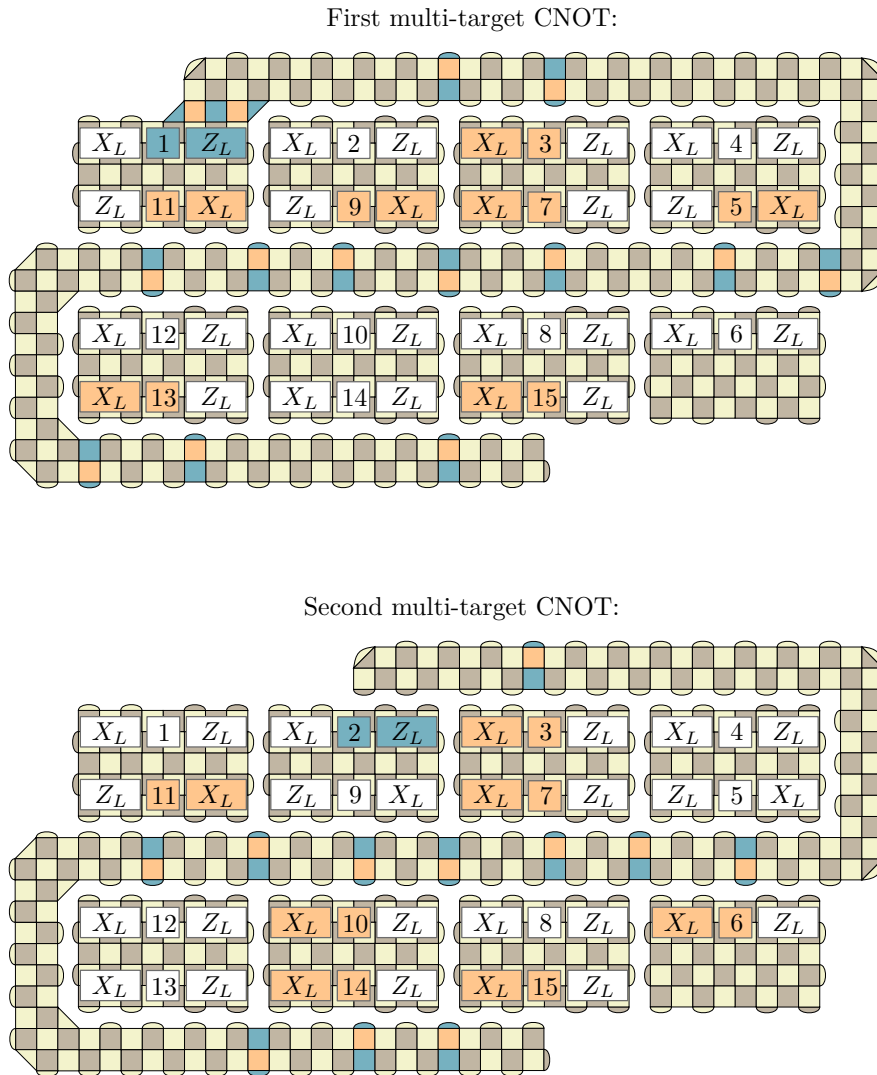


Figure 17: First and second multi-target CNOT of the distillation protocol in Fig. 10.

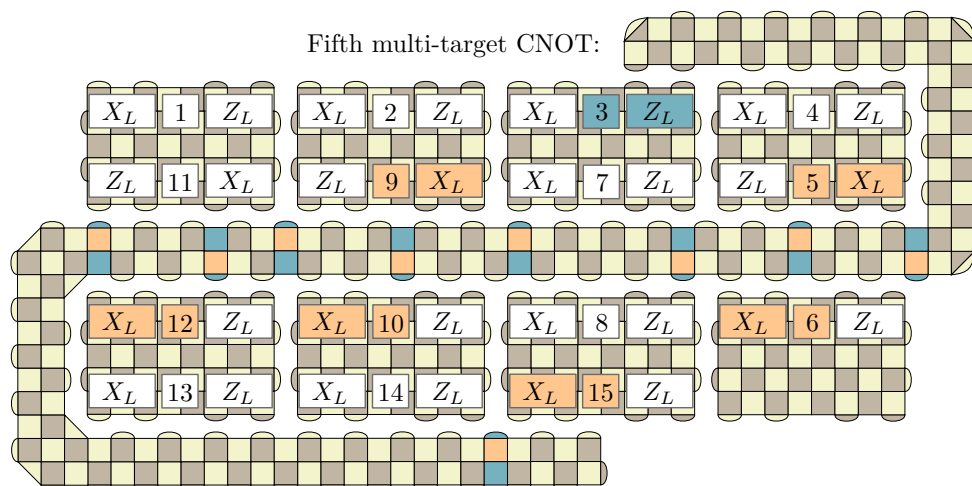
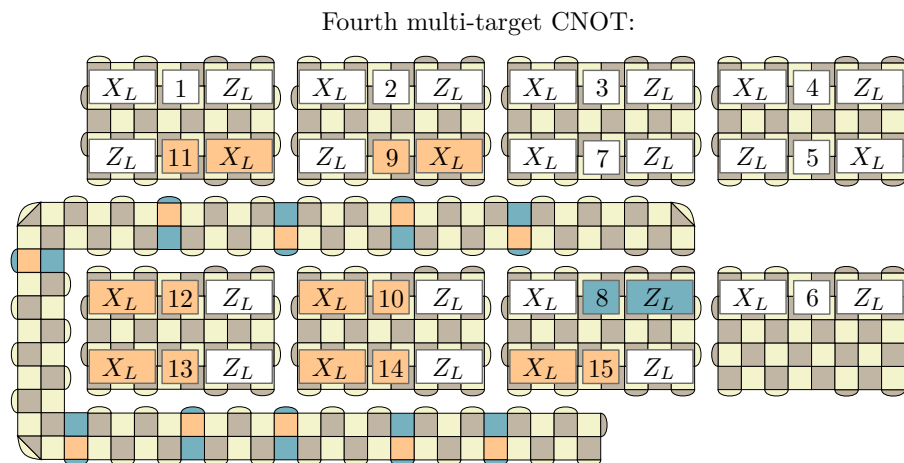
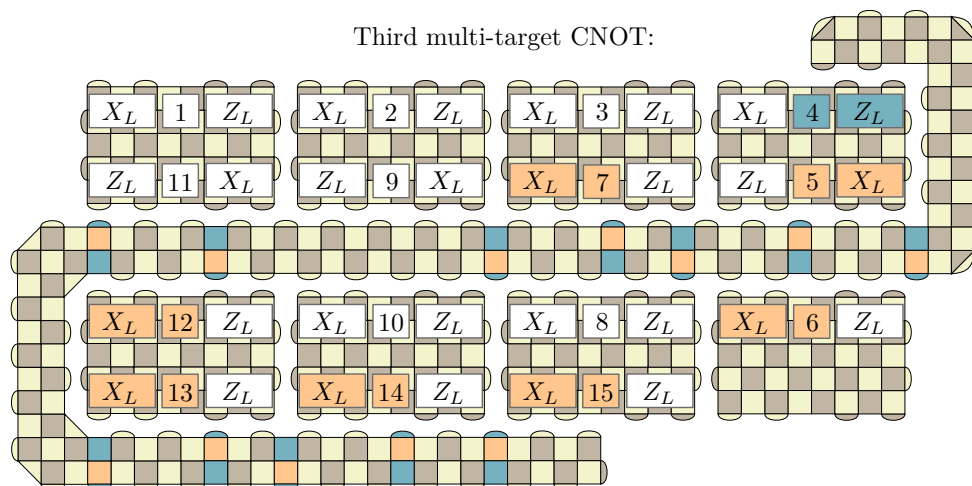


Figure 18: Third, fourth and fifth multi-target CNOT of the distillation protocol in Fig. 10.


3 | Quantum Computing with Majorana Fermion Codes

The previous chapter explained how to perform twist-based lattice surgery with qubit-based surface codes. It is worth pointing out that the 15-to-1 distillation circuit of Fig. 26 in Ref. [26] shown in Fig. 10 of the previous publication is incorrect, since the T gates need to be applied after the multi-target CNOTs, rather than before.

Qubit-based surface codes can be used in any quantum computing architecture. One of the implementations shown in the appendix of the previous chapter was for Majorana-based qubits. For error correction, these qubits are not only capable of using codes whose fundamental building blocks are physical qubits, but can also employ fermionic codes which use Majorana fermions as building blocks. In the following publication, we explore the correspondence between Majorana-based qubits and surface codes, and explain how Majorana fermion surface and color codes can be constructed and used for twist-based lattice surgery.

Quantum computing with Majorana fermion codes

Daniel Litinski and Felix von Oppen

Dahlem Center for Complex Quantum Systems and Fachbereich Physik, Freie Universität Berlin, Arnimallee 14, 14195 Berlin, Germany (Received 31 January 2018; revised manuscript received 3 April 2018; published 2 May 2018)

We establish a unified framework for Majorana-based fault-tolerant quantum computation with Majorana surface codes and Majorana color codes. All logical Clifford gates are implemented with zero-time overhead. This is done by introducing a protocol for Pauli product measurements with tetrons and hexons which only requires local 4-Majorana parity measurements. An analogous protocol is used in the fault-tolerant setting, where tetrons and hexons are replaced by Majorana surface code patches, and parity measurements are replaced by lattice surgery, still only requiring local few-Majorana parity measurements. To this end, we discuss twist defects in Majorana fermion surface codes and adapt the technique of twist-based lattice surgery to fermionic codes. Moreover, we propose a family of codes that we refer to as Majorana color codes, which are obtained by concatenating Majorana surface codes with small Majorana fermion codes. Majorana surface and color codes can be used to decrease the space overhead and stabilizer weight compared to their bosonic counterparts.

DOI: [10.1103/PhysRevB.97.205404](https://doi.org/10.1103/PhysRevB.97.205404)**I. INTRODUCTION**

In an effort to construct robust qubits for quantum computation, Majorana zero modes (Majoranas) in topological superconductors [1–6] are currently being explored as potential building blocks of topological qubits. Even though no such topological qubit has been built to date, a range of precursor experiments exist [7–10], providing various signatures of Majorana zero modes. While it remains uncertain whether Majorana-based qubits will offer higher coherence times compared to other solid-state qubits, Majorana-based qubits have one distinct feature that sets them apart from nontopological qubits. Due to the non-Abelian statistics of Majoranas, these qubits can be measured in all three Pauli bases X , Y , and Z , and not just in a single computational basis as is the case with conventional qubits. As a consequence, Majorana-based qubits implement robust single-qubit Clifford gates with zero-time overhead. These gates are products of Hadamard gates H and phase gates S , and are equivalent to the operations implemented by braiding Majoranas.

The coherence times of Majorana-based qubits are predicted to be long, but still limited [11,12]. Thus, a quantum error-correcting code is necessary for large-scale quantum computation with arbitrarily long qubit survival times. Quantum error correction combines many physical qubits into more error-resilient logical qubits [13]. Errors are detected and corrected by measuring certain stabilizer operators. Two-dimensional (2D) topological codes are of particular interest since their local stabilizers are compatible with the constraints of solid-state architectures.

Topological codes come as bosonic or fermionic codes. Bosonic codes are defined on a 2D lattice where vertices correspond to qubits and faces define stabilizers, products of Pauli operators with support on all qubits of the face. Fermionic codes are defined on a lattice where vertices correspond to Majoranas and stabilizers are products of all Majorana operators of

a face. A bosonic code maps onto a fermionic code by replacing each qubit with four Majoranas in a fixed-parity sector [14], but not all fermionic codes can be straightforwardly mapped back onto a bosonic code. This implies that Majorana-based qubits admit a wider range of topological codes than conventional qubits.

One practical problem is that fault-tolerant quantum computing requires logical operations on encoded qubits, which may be entirely different from operations on physical qubits. Indeed, various recent proposals for Majorana-based implementations of codes exist, based on, e.g., bosonic surface codes [15–18], bosonic color codes [19,20], fermionic surface codes [14,21,22], and small Majorana fermion codes [23,24], each featuring individual protocols for a universal set of logical gates. Moreover, except for twist-based encodings in surface codes [25–27] and transversal gates of bosonic color codes [19,20,28], logical single-qubit Clifford gates require a series of code operations in all aforementioned proposals, and the Majoranas' advantage of zero-overhead single-qubit Clifford gates is lost.

According to the Gottesman-Knill theorem [29], not only single-qubit Clifford gates, but all Clifford gates including the two-qubit controlled-NOT (CNOT) gate can be tracked using a classical computer. Therefore, all Clifford gates can, in principle, be implemented with zero-time overhead. Operations on logical qubits need to be appropriately designed in order to take full advantage of the Gottesman-Knill theorem.

Overview and main results

In this work, we establish a unified framework for Majorana-based quantum computation with bosonic and fermionic surface and color codes, in all of which logical Clifford gates, including CNOT gates, are implemented with zero-time overhead. To this end, we first discuss the construction of Majorana surface code patches in Sec. II. These surface

TABLE I. Comparison of different two-dimensional topological codes with respect to the following characteristics: number of Majoranas necessary to encode a logical qubit with code distance d , proposed number of Majoranas in a fixed-parity sector used as a building block (tetrons/hexons/octons/decons/dodecons refer to 4/6/8/10/12 Majoranas in a fixed-parity sector), maximum Majorana weight of the stabilizers that need to be measured, stabilizer weights in the bulk of the code (excluding the boundaries), and the maximum stabilizer weight for lattice surgery operations. For Majorana color codes, the $[[n_m, k, d_m]]$ code in parentheses is the code that is concatenated with the corresponding surface code in order to obtain the color code. The order of each code indicates how many layers of Majorana surface codes it corresponds to. All codes listed in the table implement logical Clifford gates with zero-time overhead.

Order	Code family (with minimal-overhead Clifford gates)	Majorana overhead for code distance d	Building blocks	Maximum stabilizer weight	Bulk weights	Max. stabilizer wt. for lattice surgery
Surface codes (nonoverlapping 2D local topological codes)						
1	4.6.12 Majorana surface code	$12d^2 + \mathcal{O}(d)$	dodecons	6	4, 6	6
	Bosonic subsystem surface code ^a	$12d^2 + \mathcal{O}(d)$	qubits/tetrons	6	6	10
	6.6.6 Majorana surface code ^b	$6d^2 + \mathcal{O}(d)$	hexons/octons	6	6	6
	4.8.8 Majorana surface code ^c	$4d^2 + \mathcal{O}(d)$	tetrons/hexons	8	8	8
	Bosonic surface code ^d	$4d^2 + \mathcal{O}(d)$	qubits/tetrons	8	8	10
Color codes (multiple layers of surface codes obtained by concatenation)						
2	4.8.8 ($[[6, 2, 2]]_m$) Majorana color code	$3d^2 + \mathcal{O}(d)$	hexons	8	8	8
	Bosonic 6.6.6 color code ^e	$3d^2 + \mathcal{O}(d)$	qubits/tetrons	12	12	12
3	6.6.6 ($[[20, 4, 4]]_m$) ^f Majorana color code	$2.5d^2 + \mathcal{O}(d)$	decons	12	10, 12	12
	Bosonic 4.8.8 color code ^e	$2d^2 + \mathcal{O}(d)$	qubits/tetrons	16	8, 16	12
3	4.8.8 ($[[8, 3, 2]]_m$) Majorana color code	$\frac{8}{3}d^2 + \mathcal{O}(d)$	octons	10	8, 10	10
	4.8.8 ($[[16, 3, 4]]_m$) ^f Majorana color code	$\frac{4}{3}d^2 + \mathcal{O}(d)$	octons	16	8, 16	12
4	4.8.8 ($[[20, 4, 4]]_m$) ^f Majorana color code	$1.25d^2 + \mathcal{O}(d)$	decons	16	10, 16	12
k	4.8.8 Majorana surface code	$\frac{4n_m}{kd_m^2}d^2 + \mathcal{O}(d)$		at least $4d_m$		
	Concatenated with $[[n_m, k, d_m]]_m$					
	Bosonic surface code	$\frac{4n_b}{kd_b^2}d^2 + \mathcal{O}(d)$	qubits/tetrons	at least $8d_b$		
	Concatenated with $[[n, k, d_b]]$					

First mentioned in: ^aRef. [30]; ^bRef. [21]; ^cRef. [16]; ^dRef. [31]; ^eRef. [28]; ^fRef. [24].

code patches are essentially fault-tolerant versions of tetrons and hexons which can correct a certain number of errors. While tetrons and hexons were introduced in Refs. [32,33] as quantum-wire-based [34,35] constructions, we use these terms for any physical system with four or six Majoranas in a fixed-parity sector. Next, in Sec. III, we describe a protocol to implement minimal-overhead Clifford gates with *physical* tetrons and hexons by measuring arbitrarily nonlocal Pauli product operators using only local operations. This is not fault tolerant in the sense that the protocol assumes perfect measurements and error-free qubits. However, it is entirely analogous to the fault-tolerant protocol discussed in Sec. IV, where we extend the protocol for minimal-overhead Clifford gates to *logical* tetrons and hexons, i.e., surface code patches. This is done by describing twist defects in Majorana surface codes and adapting twist-based lattice surgery [27] to fermionic codes. Finally, in Sec. V, we propose a construction procedure for Majorana color codes, i.e., the fermionic equivalent of bosonic color codes. Since bosonic color codes can be obtained by concatenating bosonic surface codes with small nontopological codes [36], we describe code concatenation for fermionic codes and construct Majorana color codes by

concatenating Majorana surface codes with small Majorana fermion codes.

With fermionic and bosonic versions of surface and color codes, a plethora of topological codes are available, all of which can implement logical Clifford gates with zero-time overhead. To decide which code to use for Majorana-based quantum computation, we show a comparison of 2D topological codes in Table I. The main differences between these codes lie in their Majorana overhead, i.e., the number of Majoranas required to encode a qubit with code distance d , and in their stabilizer weight, i.e., the number of Majorana operators contained in the stabilizers that need to be measured for error correction. It is desirable to keep both of these figures low, as a lower Majorana overhead implies a higher encoding rate and more efficient space usage, and a lower stabilizer weight implies easier implementation due to lower hardware requirements. However, the general trend seen in bosonic surface and color codes indicates that codes featuring lower-weight stabilizers tend to have a higher space overhead (or Majorana overhead in the case of Majorana-based qubits).

Out of all known bosonic 2D topological codes, surface codes have the lowest stabilizer weight of four qubits. Topological subsystem codes [37] can further reduce the weight

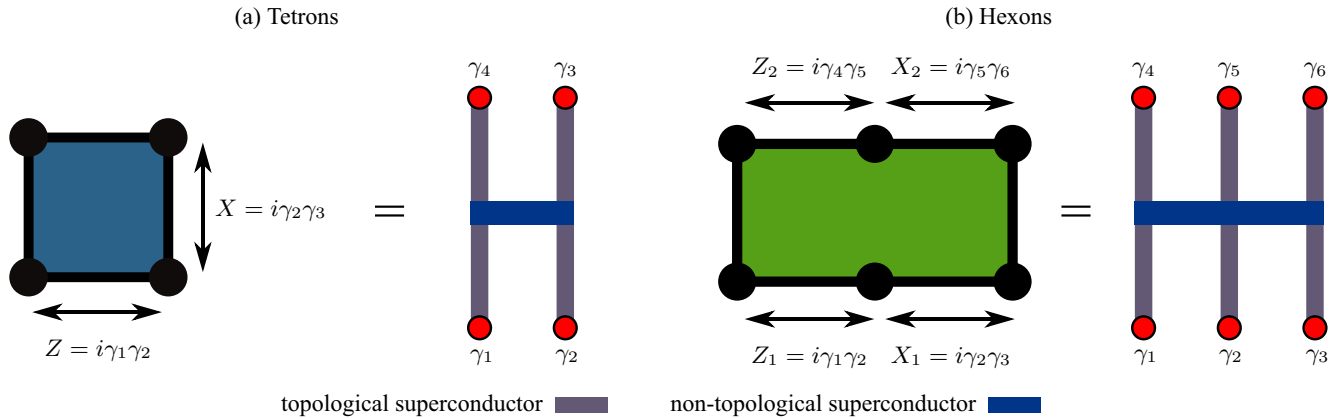


FIG. 1. Tetrons and hexons as the two smallest Majorana fermion codes, and their correspondence to 4 or 6 Majoranas in a fixed-parity sector, e.g., via the quantum-wire construction in Ref. [33].

of the operators that need to be measured, with a subsystem variant of the surface code [30] reducing the weight to three qubits. With Majorana-based hardware, this corresponds to 8-Majorana and 6-Majorana stabilizers. In order to implement minimal-overhead Clifford gates, twist-based lattice surgery requires the measurement of higher-weight five-qubit operators corresponding to twist defects, which corresponds to 10-Majorana operators. Majorana surface codes decrease the stabilizer weight below the weight of bosonic surface codes, such that only 4- and 6-Majorana operators need to be measured. This might be useful if higher-weight stabilizers turn out to be difficult to measure in a given implementation. Majorana color codes, on the other hand, can encode quantum information with a lower-space overhead compared to bosonic color codes with the same stabilizer weight.

In general, bosonic surface codes can be concatenated with any $[[n, k, d]]$ code, where n is the number of physical qubits, k is the number of encoded logical qubits, and d is the code distance, i.e., the minimum qubit weight of all logical operators. Similarly, Majorana surface codes can be concatenated with any $[[n_m, k, d_m]]_m$ Majorana fermion code, where n_m is the number of Majoranas, and d_m is the Majorana distance, i.e., the minimum Majorana weight of all logical operators. Concatenating surface codes with high-distance codes can be used to obtain topological codes with arbitrarily low space overhead. As one uses higher-distance codes for concatenation, one increasingly sacrifices stabilizer weight and locality in favor of lower Majorana overhead. From a pragmatic point of view, if a given Majorana-based quantum computer has a maximum stabilizer weight that it can measure, a comparison in the spirit of Table I can be used to determine a suitable encoding, even though we note that our collection of codes is not exhaustive. In particular, it does not include Majorana surface codes based on nonuniform tilings.

II. MAJORANA SURFACE CODES

A Majorana fermion code [14] encodes logical information in a set of Majorana zero modes placed on the vertices of 2D lattices, which are described by self-adjoint operators $\gamma_i = \gamma_i^\dagger$,

and fulfill the fermionic anticommutation relations $\{\gamma_i, \gamma_j\} = 2\delta_{i,j}$. The code is defined by its stabilizers, more precisely, by its mutually commuting stabilizer generators, which are products of all Majorana operators

$$\mathcal{O}_{\text{face}} = \prod_{j \in \text{face}} i^{1/2} \gamma_j \quad (1)$$

associated with a face and have eigenvalues ± 1 . A code with n Majoranas and m stabilizers encodes $n/2 - m$ logical qubits in the degenerate ground-state manifold of the Hamiltonian

$$H_{\text{code}} = - \sum_{i=1}^m \mathcal{O}_i. \quad (2)$$

Due to fermion parity superselection, the product of all n Majoranas is always a stabilizer, such that n Majoranas can at most encode $n/2 - 1$ qubits.

A. Smallest Majorana fermion codes

The smallest fermion code is the 4-Majorana tetron code shown in Fig. 1(a), which involves a single stabilizer $\mathcal{O}_{\text{tetron}} = -\gamma_1\gamma_2\gamma_3\gamma_4$. The qubit is encoded in the doubly degenerate ground-state space of the Hamiltonian

$$H_{\text{tetron}} = \gamma_1\gamma_2\gamma_3\gamma_4. \quad (3)$$

While it is possible to define a qubit in the Schrödinger picture by defining two computational states $|0\rangle$ and $|1\rangle$, it is more convenient to use the Heisenberg picture. Since any single-qubit unitary operator can be written using the X and Z Pauli operators via the Euler decomposition, a qubit can be defined through these two operators. The X and Z operators need to square to the identity, $X^2 = Z^2 = \mathbb{1}$, anticommute, $XZ = -ZX$, and commute with all stabilizers of the code. For a tetron, one choice is $Z = i\gamma_1\gamma_2 = i\gamma_3\gamma_4$ and $X = i\gamma_2\gamma_3 = i\gamma_1\gamma_4$. The remaining Pauli operator follows as the product $Y = iXZ$.

There are two ways to implement stabilizer terms such as $\mathcal{O}_{\text{tetron}}$. One way is to interpret stabilizers as physical parity-fixing constraints realized in the laboratory, e.g., via the charging energy of two topological superconducting nanowires.

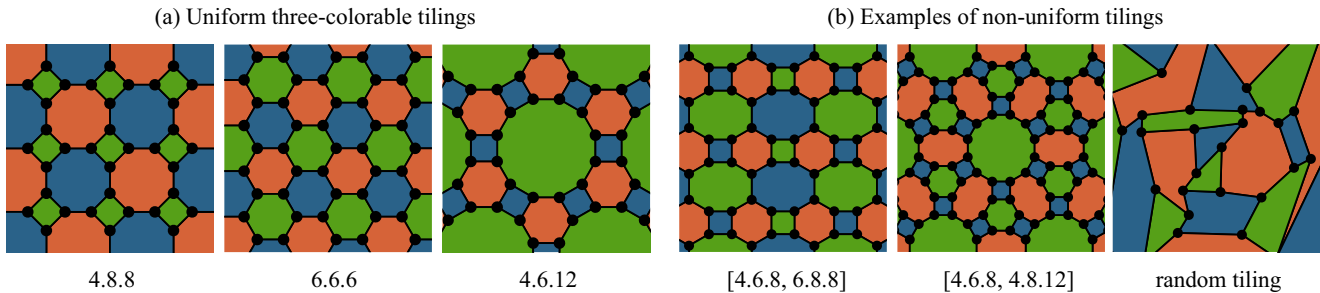


FIG. 2. The three uniform three-colorable tilings of the 2D plane (a), and three examples of nonuniform tilings (b).

Alternatively, they can be interpreted as measurement prescriptions. Measuring all stabilizers of the code projects the system into the ground-state space as long as the measurement outcome is +1 for all stabilizers. For tetrons, it is easy to see that both approaches are susceptible to errors. In particular, an error process described by the operator $i\gamma_i\gamma_j$ involving two Majorana fermion operators will lead to errors that neither violate the parity-fixing constraint nor change the measurement outcome of the parity measurement. The fact that stabilizers can be interpreted either as physical constraints or measurement prescriptions will become important when we discuss larger codes with more than one stabilizer.

The second smallest Majorana fermion code is the 6-Majorana hexon code shown in Fig. 1(b). It encodes two logical qubits in the ground-state space of the Hamiltonian

$$H_{\text{hexon}} = -i\gamma_1\gamma_2\gamma_3\gamma_4\gamma_5\gamma_6. \quad (4)$$

The logical Pauli operators of the two qubits can be chosen as $Z_1 = i\gamma_1\gamma_2$, $X_1 = i\gamma_2\gamma_3$, $Z_2 = i\gamma_4\gamma_5$, $X_2 = i\gamma_5\gamma_6$.

B. Logical tetrons and hexons

Tetrons and hexons encode qubits, but cannot correct any errors. We now describe a procedure to obtain *logical* tetrons and hexons based on 2D Majorana surface codes.

We define Majorana surface codes as fermionic topological codes with nonoverlapping stabilizers. Thus, they can be defined by a tiling of the 2D plane with vertices corresponding to Majoranas and faces (or tiles) corresponding to stabilizers. Since stabilizers need to commute, neighboring faces should share two vertices (or, in fact, any even number). This implies that any valid tiling needs to be three-colorable [14], in the sense that faces can be colored in three colors with neighboring faces having different colors. Moreover, each stabilizer needs to contain an even number of Majoranas. Remarkably, these requirements are identical to the restrictions of bosonic color codes, such that all valid color code tilings with vertices as qubits are also valid Majorana surface code tilings with vertices as Majoranas.

There exist three uniform three-colorable tilings that can be used to define a Majorana surface code: 4.8.8 [16,17,21,22], 6.6.6 [15], and 4.6.12, as shown in Fig. 2(a). The numbers refer to the three polygons that meet at each vertex, e.g., for the 4.6.12 tiling, a square, a hexagon, and a dodecagon. Uniform means that the same types of polygons meet at each vertex. There exist also nonuniform three-colorable tilings, which have at least two different vertex types. Examples are shown

in Fig. 2(b), such as the [4.6.8, 6.8.8] tiling [14] which has two types of vertices, namely, 4.6.8 and 6.8.8. In this work, we restrict ourselves to the three uniform tilings shown in Fig. 2(a), although our constructions can be straightforwardly extended to nonuniform tilings. We stress that the colors bear no physical meaning, but are merely a useful bookkeeping tool.

A logical tetron encodes one logical qubit in n Majoranas with $n/2 - 1$ stabilizers. It is constructed by appropriately introducing boundaries to a three-colorable tiling, as shown in Fig. 3. At a boundary, Majoranas are no longer part of three differently colored stabilizers, but only two. We then assign the remaining color to the boundary, e.g., on a red boundary, Majoranas are part of blue and green stabilizers. Two differently colored boundaries meet at a corner, where a Majorana is part of only one stabilizer. A logical tetron is obtained by terminating the tiling by four boundaries, a pair of opposing red and blue boundaries each (or any other pair of colors). This procedure always yields a code with n Majoranas and $n/2 - 1$ stabilizers.

One can also assign colors to the edges of faces, i.e., to the 2-Majorana operators that lie between two faces. The color of an edge is then the third remaining color. For instance, an edge that lies between a red and a blue face is referred to as a green edge. The logical X (Z) operators of a logical tetron are strings of red (blue) edges that connect the red (blue) boundaries, as shown in Fig. 3(b). In particular, the product of all Majoranas along a blue boundary is a string of red edges, i.e., an X operator. Similarly, the product of all Majoranas along a red boundary is a Z operator. Logical Z and X operators anticommute since they always share an odd number of Majoranas. The red and blue boundary operators, specifically, share one Majorana in the corner where the boundaries meet.

The four-corner surface codes depicted in Fig. 3 are indeed the higher-distance equivalents of 4-Majorana tetrons. In Ref. [38], it was pointed out that the corners of surface code qubits correspond to twist defects, which feature the same non-Abelian statistics as Majoranas [25]. This can be understood from the observation that since one type of surface code boundary can absorb e anyons, and the other type of boundary can absorb m anyons, corners can absorb ϵ anyons. Therefore, according to Ref. [25], corners are twist defects. Alternatively, in Ref. [39], edges of surface codes are identified as flat bands of uncoupled Majoranas. The boundary stabilizers gap out the edge Majoranas, leaving unpaired Majoranas in the corners. Thus, the identification of four boundary Majorana surface codes with tetrons is indeed justified, as their corners can be interpreted as *logical* Majoranas, i.e., twist defects.

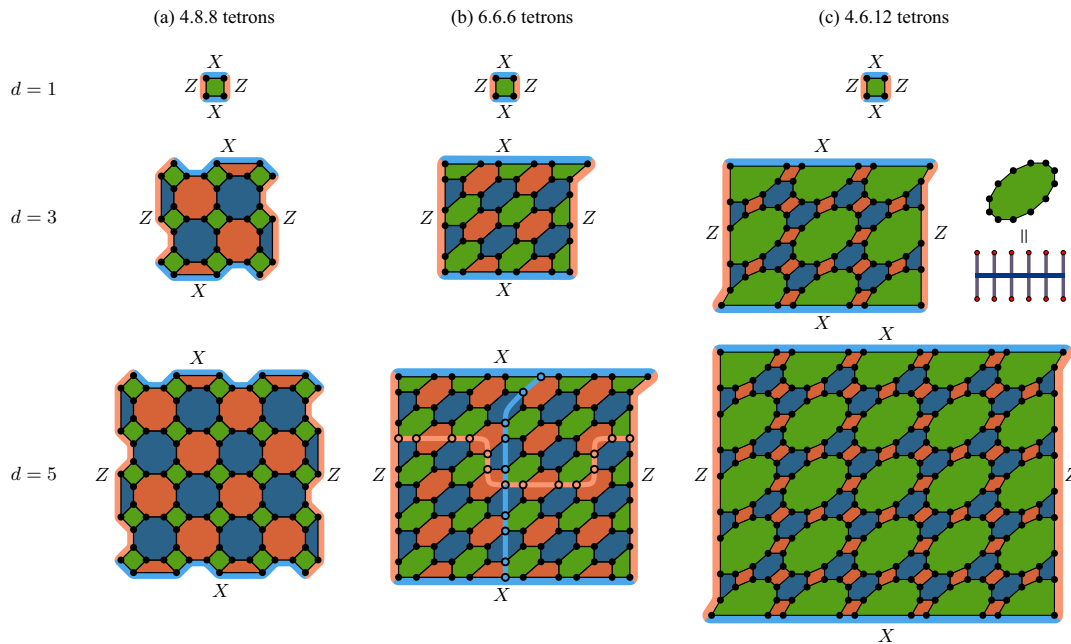


FIG. 3. Logical tetrons of the 4.8.8 (a), 6.6.6 (b), and 4.6.12 (c) Majorana surface code with code distances $d = 1, 3$, and 5 . The 6.6.6 Majorana surface code with $d = 5$ shows an example of strings of red and blue edges through the bulk of the code. Products of Majoranas along the red and blue strings are equivalent to logical X and Z operators, respectively.

Similarly, six boundary Majorana surface codes can be identified as logical hexons, as shown in Fig. 4. Logical hexons encode two logical qubits on n Majoranas with $n/2 - 2$ stabilizers. This can be achieved by introducing three red and blue boundaries each in an alternating fashion. Logical hexons require approximately twice as many Majoranas as logical tetrons for a given code distance, but also encode twice as many qubits, one qubit at the two bottom and one qubit at the two top boundaries.

The error processes considered in this work correspond to quasiparticle poisoning and are described by the action of single Majorana operators γ_i . We refer to these errors as Majorana errors. A Majorana error flips the measurement outcome of the stabilizers that the affected Majorana is part of, which allows for detection of errors and their subsequent correction. Corrections do not need to be applied explicitly, but merely tracked by a classical computer. We define the Majorana code distance d_m as the minimum number of Majorana errors necessary to introduce a logical error that will go undetected, i.e., d_m is the number of Majorana operators contained in the shortest logical operator. Codes with Majorana distance d_m tolerate $d_m/2 - 1$ Majorana errors, as the syndrome of error strings that cover more than half the Majorana distance will be misinterpreted and lead to a correction that introduces a logical Pauli error. Since only even-number products of Majoranas are physically measurable, d_m is always even. For this reason, d_m is not the same quantity as the code distance d of bosonic codes. For a fair comparison between fermionic and bosonic codes, we label fermionic codes by their bosonic code distance $d = d_m/2$. This is justified because Pauli operators of physical Majorana-based qubits, such as tetrons or hexons, correspond to products of two Majorana operators, and therefore single-qubit Pauli errors always correspond to *two* Majorana errors. Thus, the

Majorana surface code tetrons shown in Fig. 3 have Majorana distances $d_m = 2, 6$, and 10 , but code distances $d = 1, 3$, and 5 .

To minimize the number of operators that need to be measured, we implement stabilizers of one color as parity-fixing constraints, while the other two colors are interpreted as measurement prescriptions. The parity-fixed color should be the one color that is not a boundary. For instance, the green 4-Majorana stabilizers of the 4.8.8 Majorana surface codes in Fig. 3(a) could be interpreted as physical tetrons, whereas the red and blue stabilizers are operators that need to be measured in order to actively detect and correct errors. This can be done with any three-colorable tiling, such that, say, red and blue stabilizers need to be measured, while green stabilizers are interpreted as Majoranas in a fixed-parity sector. We then refer to the green stabilizers as the building blocks of the code. Note that Majorana errors that respect the parity-fixing constraints set by the green stabilizers always come in pairs.

If green stabilizers are not measured, but implemented as parity-fixing constraints, then single-Majorana errors violating these constraints are leakage errors, i.e., errors that take the qubit out of the computational subspace and may go undetected by the measurement of red and blue stabilizers. For quantum-wire-based architectures, the timescales of dephasing and depolarizing errors that are detectable by red and blue stabilizer measurements were calculated to range from hundreds of nanoseconds to several minutes for achievable experimental parameters [12]. On the other hand, leakage errors caused by single-Majorana errors are suppressed exponentially with $\exp(-E_C/T)$, where E_C is the charging energy fixing the parity, and T is the temperature. Already for $E_C \sim 100 \mu\text{eV}$ and $T \sim 10 \text{ mK}$, $E_C/T \approx 100$, which suggests that leakage may be negligible [12]. In implementations where leakage errors

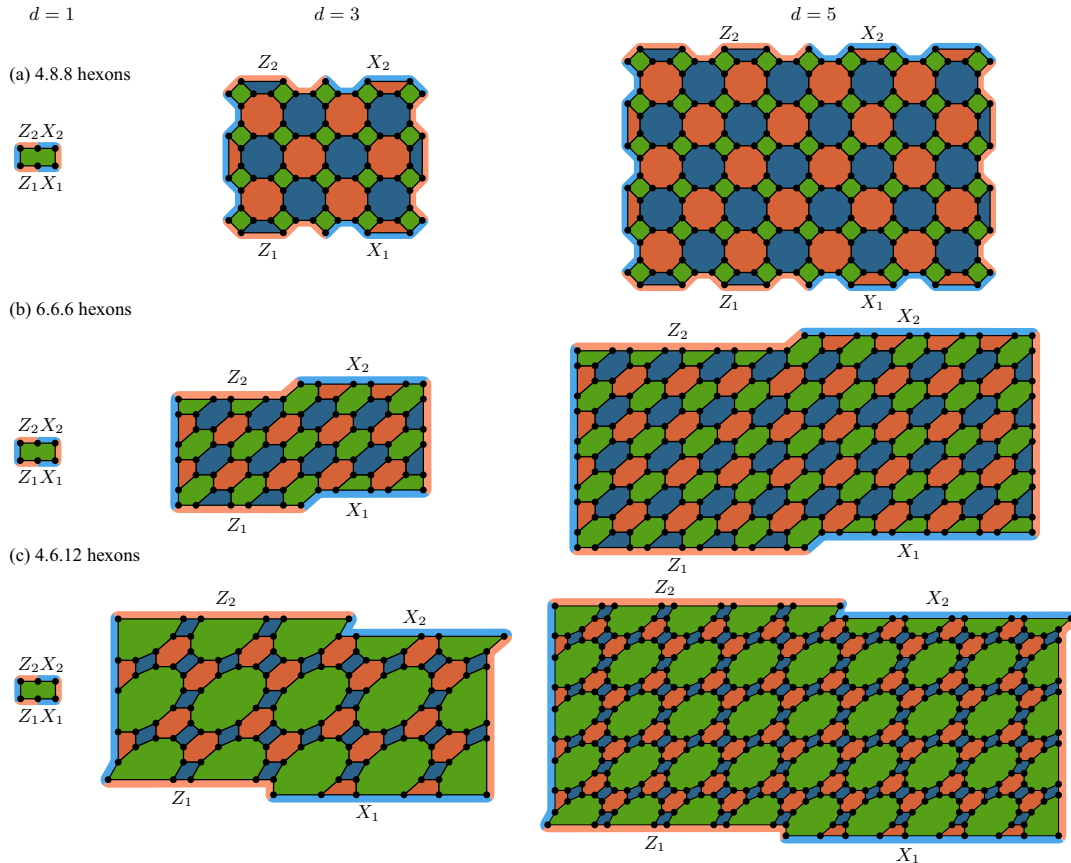


FIG. 4. Logical hexons of the 4.8.8 (a), 6.6.6 (b), and 4.6.12 (c) Majorana surface code with code distances $d = 1, 3$, and 5 .

cannot be neglected, they can be treated by also measuring the green stabilizers.

4.8.8 Majorana surface codes can be defined on a square lattice of tetrons in the spirit of Ref. [16], such that the 8-Majorana stabilizers need to be measured. One qubit with code distance d requires $4d^2$ Majoranas. Similarly, 6.6.6 Majorana surface codes can be defined on a square lattice of hexons. While they require more Majoranas per qubit ($6d^2 - 4d + 2$), the weight of the operators that needs to be measured reduces to 6 Majoranas. We note that even though the codes in Fig. 3(b) display green tetrons at the top and right boundaries, these codes can indeed be defined on a square lattice of hexons, as we show in Appendix A 1. Finally, 4.6.12 Majorana surface codes can be defined on a square lattice of dodecons, i.e., 12 Majoranas in a fixed-parity sector. In the context of color codes, 4.6.12 codes are often ignored since they feature both a high-space overhead and high-weight stabilizers. However, by interpreting the 12-Majorana operators as building blocks that satisfy parity-fixing constraints, only 4- and 6-Majorana operators need to be measured. Thus, the stabilizer weight has decreased to an average of 5 Majoranas, albeit at the price of an increased space overhead of $12d^2 - 12d + 4$ Majoranas per qubit.

An important part of error correction is decoding. While the error syndrome is known from the stabilizer measurements, the most likely error configuration that causes this syndrome needs to be determined by a classical program called a decoder. Since Majorana surface codes are indeed surface codes, they

can be decoded by any bosonic surface code decoder,¹ either by interpreting blue and red stabilizers as two types of anyons that need to be matched (analogous to e and m anyons in bosonic surface codes), or by first mapping the Majorana code onto a bosonic code and then decoding the bosonic surface code. As we show in Appendix A 2, Majorana surface codes can be mapped onto bosonic codes, but the mapping is not unique. In the case of 4.8.8 codes, replacing each tetron with a qubit yields exactly the bosonic surface code on a square lattice. Replacing each hexon of the 6.6.6 code with two qubits also yields a bosonic code on a square lattice, but the lattice is rotated. With a 4.6.12 code, replacing each of the 4-Majorana operators with a qubit yields a bosonic surface code on a trihexagonal lattice.

III. MINIMAL-OVERHEAD CLIFFORD GATES WITH TETRONS AND HEXONS

In this section, we show how to perform universal quantum computation with *physical* tetrons and hexons. This is not fault tolerant since the protocols assume perfect measurements and error-free qubits. Closely analogous fault-tolerant protocols

¹We note that this is only true if building blocks are interpreted as parity fixed, such that, say, green stabilizers cannot be violated. If this is not the case, then the decoder needs to match all three colors, similar to the decoding of bosonic color codes.

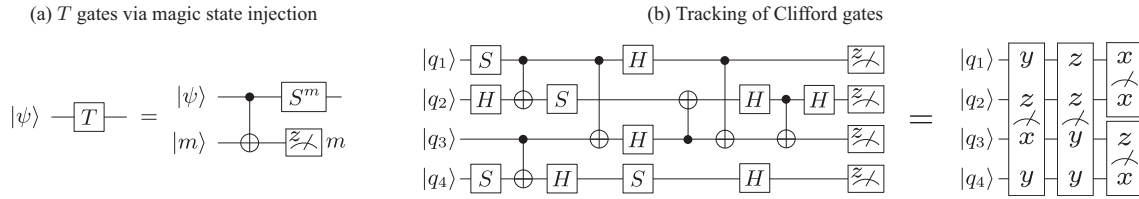


FIG. 5. (a) Applying a T gate to a qubit $|\psi\rangle$ is equivalent to a CNOT between the qubit and a magic state $|m\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)$, a measurement of the magic state with outcome $m \in \{0, 1\}$, and a corrective operation S^m on the qubit. (b) Example of a circuit of Clifford gates and four Z measurements that is compressed into four Pauli product measurements.

based on logical tetrons and hexons will be discussed in Sec. IV.

We start with general considerations concerning the classical tracking of Clifford gates. These gates map Pauli operators onto other Pauli operators and are products of Hadamard gates H , phase gates S , and controlled-NOT gates CNOT. Any quantum circuit can be expressed in terms of Clifford gates, T gates (where $T = \sqrt{S}$), and single-qubit measurements in the Z basis [40]. If magic states $|m\rangle = (|0\rangle + e^{i\pi/4}|1\rangle)/\sqrt{2}$ are available as a resource, T gates can be rewritten as a CNOT gate and a measurement of the magic-state qubit [see Fig. 5(a)]. In realistic architectures, only faulty magic states are available. Using magic-state distillation [41], multiple faulty magic states can be converted into fewer low-error magic states. This is yet another circuit of Clifford gates and measurements. Thus, any quantum computation can be expressed as a quantum circuit consisting only of Clifford gates and measurements.

According to the Gottesman-Knill theorem, gate operations for Clifford gates need not be performed, as they can be tracked classically. Mapping Pauli operators onto other Pauli operators, they merely change the basis of the Z measurements: H gates map $Z \rightarrow X$ and $X \rightarrow Z$, S gates map $Z \rightarrow Z$ and $X \rightarrow Y$, and CNOTs map $\mathbb{1} \otimes Z \rightarrow Z \otimes Z$, $Z \otimes \mathbb{1} \rightarrow Z \otimes \mathbb{1}$, $X \otimes \mathbb{1} \rightarrow X \otimes X$, and $\mathbb{1} \otimes X \rightarrow \mathbb{1} \otimes X$. The action of Clifford gates on all other Pauli operators can be inferred from these rules. Consequently, any circuit of Clifford gates and n measurements can be contracted to n measurements of Pauli product operators. Thus, apart from the generation of (faulty) magic states, a quantum computer merely needs to be able to fault tolerantly measure nonlocal products of Pauli operators. An example of such a contraction is shown in Fig. 5(b). In this sense, a quantum computer that can measure arbitrary Pauli products implements Clifford gates with zero-time overhead, as these gates can be tracked classically and do not require any quantum operations.

The state-injection circuit in Fig. 5(a) then corresponds to a $Z \otimes Z$ measurement between the qubit $|\psi\rangle$ and the magic state, and a tracked S -gate correction. In order to discard the magic-state qubit, it needs to be disentangled via an X measurement and a subsequent Z correction on the qubit $|\psi\rangle$, as we discuss in Appendix A 3. We note that if one prefers to explicitly perform CNOT operations, this can also be done using Pauli product measurements via the circuit identity shown in Appendix A 3.

Pauli product measurements with tetrons and hexons

We now describe a protocol for Pauli product measurements with tetrons and hexons using only local two-qubit operations.

Our scheme uses hexons to encode qubits used for quantum computation (referred to as data qubits), and tetrons to encode ancilla qubits that enable long-range communication. The hexons form blocks of data qubits in a 2D array, with tetrons as ancillas between blocks. A possible arrangement of 24 data qubits is shown in Fig. 6. We describe the protocol using the example in Fig. 7.

The goal is to measure the nonlocal n -qubit Pauli product $Z_1 \otimes X_3 \otimes Z_8 \otimes Y_9$ without measuring any of the n individual Pauli operators Z_1, X_3, Z_8 , or Y_9 . The first step is to initialize an $n + m$ -ancilla Greenberger-Horne-Zeilinger (GHZ) state $|0\rangle^{\otimes n+m} + |1\rangle^{\otimes n+m}$, where m ancillas are used to bridge long distances. In our example, we use $n + m = 4 + 2$ ancilla qubits. Each ancilla is measured in the X basis by measuring the corresponding 2-Majorana operator (labeled $X_{a1} \dots X_{a6}$) and thereby prepared in the X eigenstate $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$. For measurement outcomes -1 , a corrective Z operation is necessary, which can also be tracked. Measuring $Z \otimes Z$ of neighboring pairs of $|+\rangle$ ancillas (corresponding to the red 4-Majorana operators shown in step 2 of Fig. 7) prepares the ancillas in the desired GHZ state, assuming that all $Z \otimes Z$ measurements yield $+1$. Again, measurement outcomes -1 require corrective Pauli X operations.

The GHZ state is an entangled state with known global properties, but unknown local properties. It is a $+1$ eigenstate of $\tilde{X} = X^{\otimes n+m}$, but the measurement outcome of each individual X operator is entirely random. Thus, we can use the GHZ state to measure the desired Pauli product $\tilde{X} \otimes Z_1 \otimes X_3 \otimes Z_8 \otimes Y_9$ without measuring any of the individual operators. By measuring the six operators shown in step 3 of Fig. 7, the desired operator is obtained as the product of the six measurements.

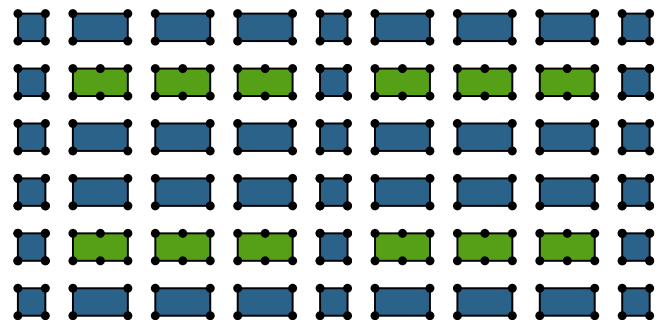


FIG. 6. Array of tetrons and hexons, where hexons are used to encode data qubits, and tetrons are used as ancillas for Pauli product measurements.

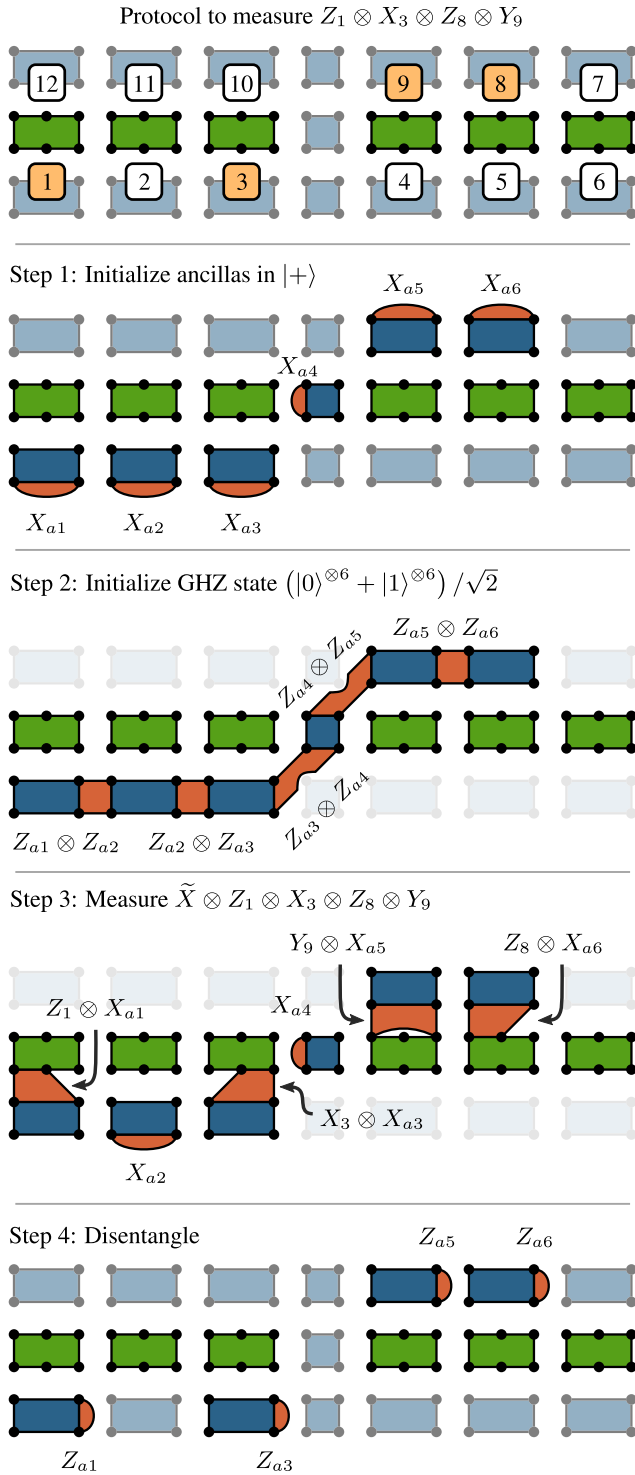


FIG. 7. Protocol for the measurement of the Pauli product operator $Z_1 \otimes X_3 \otimes Z_8 \otimes Y_9$.

In the general case of the measurement of a Pauli product $P_1 \otimes \cdots \otimes P_n$, the n ancilla qubits adjacent to the corresponding data qubits are used to measure $P_n \otimes X_{a,n}$, while the remaining m ancilla qubits are read out in the X basis. The product of all $n + m$ measurements is $\tilde{X} \otimes P_1 \otimes \cdots \otimes P_n$. Because $\tilde{X} = 1$, but each individual $X_{a,n}$ is random, this

measures the Pauli product without measuring any of the individual Pauli operators.

The n ancilla qubits adjacent to data qubits still need to be disentangled from the data qubits in step 4 of the protocol, before they can be discarded. This is done by measuring the ancilla qubits in the Z basis with outcome $m \in \{0, 1\}$, leading to a P^m Pauli correction on the adjacent data qubit, where P is the Pauli operator that was part of the two-qubit measurement in step 3, e.g., Z_1 for qubit 1 in Fig. 7.

When combined with the preparation of (faulty) magic states, this is sufficient to implement universal quantum computation. Specifically, this protocol can be straightforwardly used with the quantum-wire constructions of Ref. [33]. In Appendix B 1, we explicitly show the tunnel-coupling configurations to implement Fig. 7 in a network of topological superconducting nanowires.

To summarize, the following operations are necessary to implement a Majorana-based universal quantum computer with minimal-overhead Clifford gates:

- (Op1) measurements of tetrons in the bases X and Z ;
- (Op2) measurements of $Z \otimes Z$ between adjacent tetrons;
- (Op3) measurements of $X \otimes X$, $Y \otimes X$, and $Z \otimes X$ between adjacent hexons and tetrons;
- (Op4) application of (potentially faulty) T gates on tetron qubits.

In the following section, we show how to implement these four operations fault tolerantly with *logical* tetrons and hexons.

IV. TWISTS IN MAJORANA SURFACE CODES

Remarkably, a completely analogous scheme can be implemented with logical tetrons and hexons, allowing for fault-tolerant quantum computing with tracked Clifford gates. Specifically, we implement the four operations (Op1)–(Op4) with the logical tetrons and hexons from Figs. 3 and 4. The first operation is the measurement of logical tetrons in the X and Z bases. With bosonic surface codes, logical qubits are read out in the X or Z basis by measuring all physical qubits in the X or Z basis, and performing classical error correction. Similarly, Majorana surface code tetrons are read out in the X or Z basis by measuring all 2-Majorana operators corresponding to red or blue edges, and performing classical error correction. The measurement outcome corresponds to the product of all Majoranas along the corresponding boundary of the tetron. In analogy to bosonic surface codes, Majorana surface code tetrons can be initialized in the $|+\rangle$ state by measuring all red edges, and then measuring the stabilizers and correcting the errors.

The second and third operations require measurements of two-qubit Pauli products. For logical tetrons with code distance d , this corresponds to products of $4d$ Majoranas, which are highly nonlocal operators. In order to measure these operators using only measurements of low-weight local operators, we adapt the technique of lattice surgery [42], and in particular twist-based lattice surgery [27], to fermionic codes. While we only explicitly show lattice surgery protocols for the three uniform Majorana surface codes, similar constructions can also be used for nonuniform tilings.

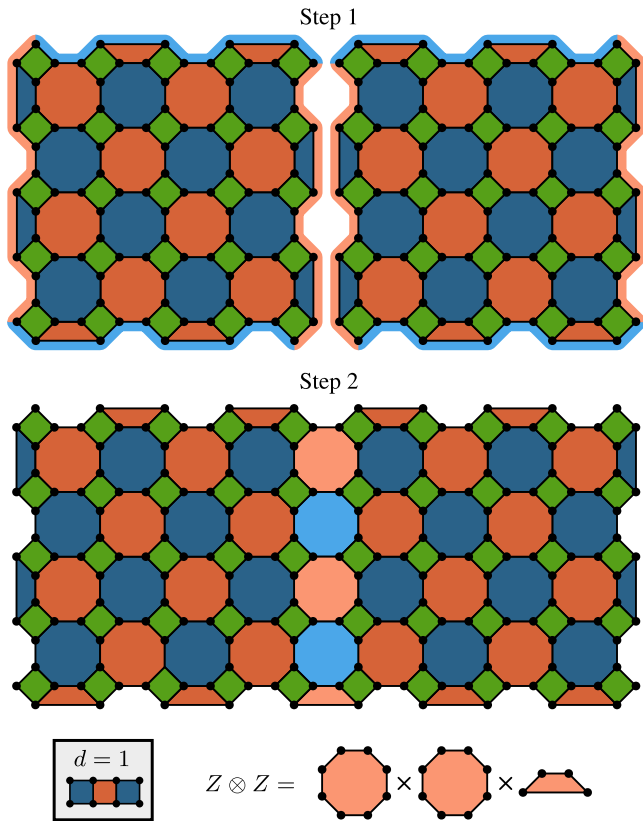


FIG. 8. Lattice surgery between two 4.8.8 Majorana surface code tetrons with code distance $d = 5$ measuring $Z \otimes Z$. The box in the bottom left corner shows the corresponding measurement with $d = 1$, where $Z \otimes Z$ is the red 4-Majorana operator.

A. 4.8.8 Majorana surface codes

We start with 4.8.8 Majorana surface codes. The second operation (Op2) requires the measurement of $Z \otimes Z$ between two tetrons, such as the two $d = 5$ tetrons in Fig. 8. In this example, the operator $Z \otimes Z$ is the product of the 20 Majorana fermions along the two neighboring red boundaries. Lattice surgery is a protocol that temporarily changes the stabilizer configuration in order to measure this operator fault tolerantly using only local measurements. Pairs of blue 4-Majorana stabilizers are merged to form 8-Majorana stabilizers (light blue in step 2). The measurement outcome of these 8-Majorana stabilizers is trivial, as it is simply the product of previously known stabilizers. In addition, new stabilizers are introduced along the boundary (light red). In this new stabilizer configuration in step 2, all stabilizers commute, but the number of stabilizers is increased by one compared to step 1. Thus, the number of degrees of freedom is reduced by one, and one bit of information is measured. Since the light blue stabilizers are trivial, this bit of information is given by the product of the light red stabilizers. Because the stabilizers involve each red-boundary Majorana exactly once, their product is exactly $Z \otimes Z$. This is the fault-tolerant generalization of the 4-Majorana parity measurement between two physical tetrons (with $d = 1$) that is shown in the inset in Fig. 8. Not only is this protocol fault tolerant in the sense that it can correct qubit errors such as quasiparticle poisoning, but by repeating rounds

of syndrome measurement, erroneous stabilizer measurements can also be corrected.

A very similar protocol can be used for the third operation to measure $X \otimes X$, $Y \otimes X$, and $Z \otimes X$ between a tetron and a hexon. In Fig. 9(a), an $X \otimes X$ measurement is shown for $d = 5$. Here, $X \otimes X$ is the product of the 20 Majoranas along the two adjacent blue boundaries. Again, the 4-Majorana operators are merged to 8-Majorana operators (light red), and the product of the new light blue operators yields precisely $X \otimes X$. The protocol for the $Z \otimes X$ measurement in Fig. 9(b) measures the product of 20 Majoranas that are located on a red boundary on the hexon, but on a blue boundary on the tetron. While again stabilizers are merged and new stabilizers are introduced to yield $Z \otimes X$, the stabilizer configuration is different from the situation in Fig. 9(a). Because the two boundaries have different colors, the stabilizers in step 2 form what is called a *dislocation line*. If flipped red and blue stabilizers are interpreted as anyons in analogy to e and m anyons of bosonic surface codes, then anyons passing this dislocation line will change color.

The measurement of $Y \otimes X$ is shown in Fig. 9(c). Because $Y = iXZ$, the Y operator of the (bottom) hexon qubit is the product of 18 Majoranas on both the red and blue bottom boundaries, excluding the center Majorana where the two boundaries meet (since it is part of both boundaries and $\gamma^2 = 1$). Thus, $Y \otimes X$ is the product of 36 Majoranas on the long blue boundary of the tetron, and the adjacent red and blue boundaries of the hexon. The product is again measured by merging stabilizers and introducing new ones whose product is precisely the product of the 36 Majoranas. The stabilizer configuration in step 2 corresponds to a dislocation line that is terminated by a 10-Majorana stabilizer (orange). This 10-Majorana stabilizer is what is called a twist defect [25]. Twists are found at the ends of dislocation lines. Incidentally, corners of surface code qubits, i.e., the meeting points of two different boundaries, can also be interpreted as ends of dislocation lines, and therefore as twists [38]. The protocols in Figs. 8 and 9 further illustrate the equivalence between twists and Majoranas: What for $d = 1$ is a measurement of a product of 4 Majoranas, for higher code distances becomes a measurement of the four twists in the corners that are part of the lattice surgery protocols. Twist defects become visible when the twist is not located in a corner during lattice surgery. This is what happens in Fig. 9(c) for the $Y \otimes Z$ measurement, where the 10-Majorana twist defect is revealed as it is in the center of the qubit corresponding to the 1 Majorana that is not part of the parity measurement in the $d = 1$ case.

The 10-Majorana twist defect of the 4.8.8 Majorana surface code corresponds to the five-qubit twist defect of bosonic surface codes. What is more, all of the aforementioned lattice surgery protocols with 4.8.8 Majorana surface codes are exactly equivalent to the protocols with bosonic surface codes [27]. In fact, no Majorana fermion code that is purely based on tetron building blocks (such as 4.8.8 codes) can display any features that are different from bosonic codes, as any bosonic code can be mapped onto a fermionic code by replacing each qubit with a tetron [14]. However, by replacing only a few of the tetrons with hexons, the 4.8.8 Majorana surface code can display some Majorana-specific characteristics. In particular, the maximum stabilizer weight necessary for lattice surgery

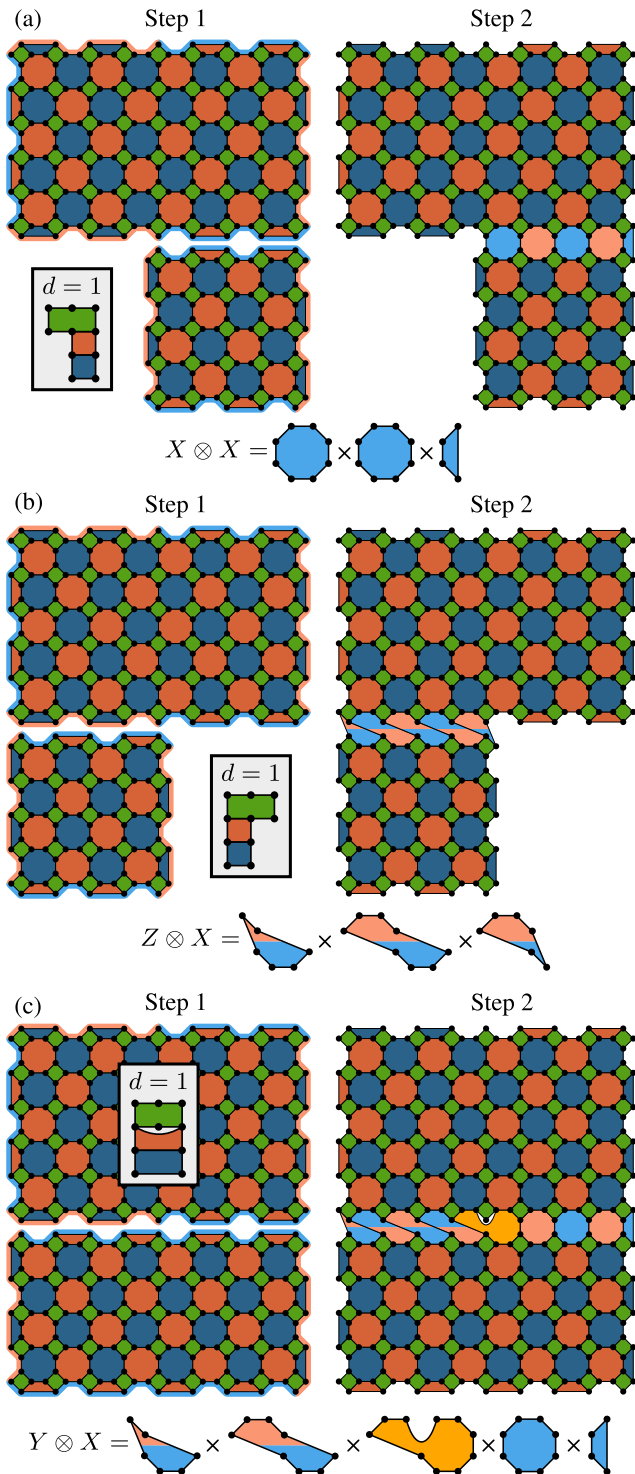


FIG. 9. Lattice surgery protocols for the measurements of the operators $X \otimes X$ (a), $Z \otimes X$ (b), and $Y \otimes X$ (c) between a 4.8.8 Majorana surface code hexon and tetron. The insets show the equivalent operation with $d = 1$.

can be reduced from 10 Majoranas (corresponding to five qubits in the bosonic case) to 8 Majoranas.

Consider the modified $d = 5$ hexons shown in Fig. 10. The tetrons along the bottom and top red boundaries have been

replaced by hexons. Since two of the Majoranas of the hexons are in a fixed-parity sector due to the 2-Majorana plaquettes, they are equivalent to tetrons. In order to measure $Y \otimes X$ between the hexon and a tetron via lattice surgery, stabilizers are merged and new stabilizers are introduced to yield the desired operator. The stabilizer configuration again features a dislocation line that is terminated by a twist defect, but the weight of the twist defect has decreased to just 8 Majoranas. Due to the three-colorability of Majorana surface code lattices, their twist defects closely resemble the so-called color twists of bosonic color codes [43].

Thus, 4.8.8 Majorana surface codes feature the same Majorana overhead as bosonic surface codes of $\sim 4d^2$ Majoranas per qubit, but a lower stabilizer weight for twist-based lattice surgery. As we show in the remainder of the section, 6.6.6 and 4.6.12 Majorana surface codes can be used to reduce the stabilizer weight even further, albeit at the cost of a higher Majorana overhead.

Note that, in contrast to the Pauli product measurement protocol for physical tetrons, the fault-tolerant protocol features tetrons of different lengths. Specifically, the tetrons used for $Y \otimes X$ measurements are almost twice as long as tetrons used for $X \otimes X$ or $Z \otimes X$ measurements. The protocol of Fig. 7 can still be used in the fault-tolerant setting, but additional tetrons may be required to bridge distances between ancillas. For concreteness, we show an implementation of the protocol of Fig. 7 using 4.8.8 Majorana surface codes with $d = 3$ in Appendix A4.

B. 6.6.6 and 4.6.12 Majorana surface codes

6.6.6 codes only require the measurement of 6-Majorana operators in the bulk of the code, as opposed to 8-Majorana operators for 4.8.8 codes. In Fig. 11(a), we show the protocol to measure $Y \otimes X$ between a 6.6.6 hexon and tetron. This case is the most instructive, as it features standard lattice surgery, a dislocation line, and a twist defect. The measurement protocols for the other operators are shown in Appendix A5. Even though 6.6.6 codes reduce the stabilizer weight to 6 Majoranas for most stabilizers, twist-based lattice surgery requires the measurement of some 8-Majorana operators that are part of dislocation lines. However, we can use the same trick as for 4.8.8 Majorana surface codes to reduce the weight of the lattice surgery operators. As we show in Appendix A6, the weight of the dislocation line operators can be reduced to 6 Majoranas by replacing some of the boundary hexons with octons (8 Majoranas in a fixed-parity sector), such that at most 6-Majorana parity measurements are required for quantum computing with 6.6.6 Majorana surface codes.

Figure 11(b) shows the same scenario for 4.6.12 codes. Here, both the dislocation line and the twist defect consist of operators that have a weight of six Majoranas. While the green building blocks along the dislocation line are drawn as decons (10-Majorana building blocks) in Fig. 11(b), as opposed to dodecons, this protocol can in fact be realized in a square lattice of dodecons. This is shown in Appendix A5, where we also present the remaining lattice surgery protocols with 4.6.12 codes.

Thus, 4.6.12 Majorana surface codes require at most 6-Majorana parity measurements for quantum computation.

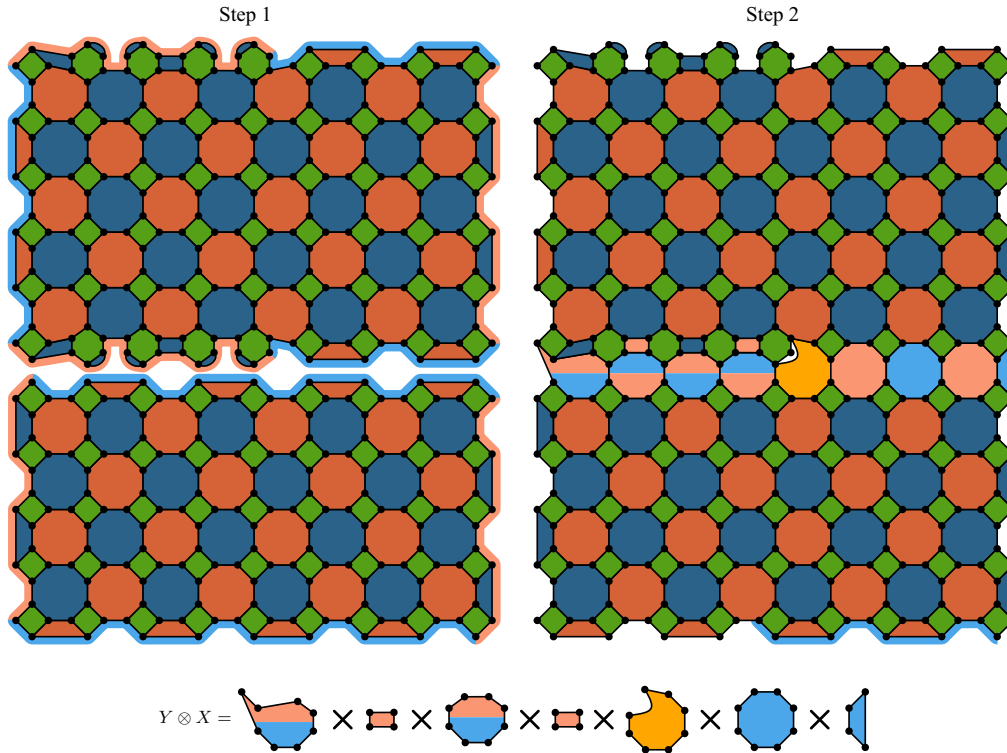


FIG. 10. Lattice surgery protocol for the measurement of $Y \otimes X$ between a 4.8.8 Majorana surface code hexon and tetron. The hexon is modified to feature hexons instead of tetrons at the bottom and top red boundaries. As a consequence, the weight of the twist defect (orange) reduces to 8 Majoranas.

Compared to bosonic surface codes, this comes at the price of a threefold increased space overhead of $\sim 12d^2$ Majoranas per qubit. This is reminiscent of bosonic subsystem surface codes [30], which only require three-qubit operator measurements (corresponding to 6 Majoranas), and also feature an increased space overhead of $\sim 4d^2$ physical qubits (corresponding to $\sim 12d^2$ Majoranas) per logical qubit. However, since these are bosonic codes, twist-based lattice surgery features a five-qubit twist defect, corresponding to 10-Majorana operators. Since there exists no work on lattice surgery with these codes, we show the twist-based lattice surgery protocol with subsystem surface codes in Appendix A 7, even though it is exactly the same as with standard bosonic surface codes.

While all of our Majorana surface code constructions can be applied to any Majorana platform, they can in particular be implemented using the quantum-wire constructions of Ref. [33]. We show nanowire implementation of the three uniform Majorana surface codes in Appendix B 2.

C. State injection

So far, we have shown how to implement the operations (Op1), (Op2), and (Op3) with 4.8.8, 6.6.6, and 4.6.12 Majorana surface codes, which allow tracking of Clifford gates, thus implementing them with zero-time overhead. The remaining operation required for universality is the generation of (faulty) magic states that are encoded in tetrons. Unfortunately, even under the assumption of error-free $d = 1$ tetrons, there is no robust way to initialize a magic state. One possibility to prepare

a faulty magic state is to initialize a $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ state by measuring $i\gamma_2\gamma_3$, and to split the degeneracy between the two states $|0\rangle$ and $|1\rangle$ by ΔE for a time $\tau = \pi/4 \cdot \hbar/\Delta E$. The dynamical phase difference between the states $|0\rangle$ and $|1\rangle$ will yield a magic state $|m\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)$. The degeneracy splitting can be implemented by coupling the two Majoranas γ_1 and γ_2 , as is done for the measurement of $Z = i\gamma_1\gamma_2$. This protocol is obviously not robust against perturbations, as it requires precise control of the coupling parameters to generate an exact phase difference of $\pi/4$. While there exist more sophisticated protocols for $\pi/8$ rotations in Majorana-based architectures [44–46], magic-state distillation is still required to obtain a low-error magic state.

Majorana magic gates or the coupling of two Majoranas can be used to obtain a noisy magic state encoded in a ($d = 1$) tetron. Fault-tolerant quantum computing requires us to convert this magic state into a magic state encoded in a *logical* tetron with a higher code distance. This is done via a protocol called state injection. There exist several protocols to inject an arbitrary state into a surface code [42,47,48]. In particular, Ref. [47] describes a two-step protocol that minimizes the time that the magic state spends with code distance $d = 1$. Here, we adapt this protocol to fermionic codes.

Consider the stabilizer configuration shown in the top panel of Fig. 12(a). The number of stabilizers is the same as in a 4.8.8 Majorana surface code with $d = 5$, i.e., exactly one qubit is encoded. Even though the four segments in the corners look like small surface codes, they do not encode any qubits since

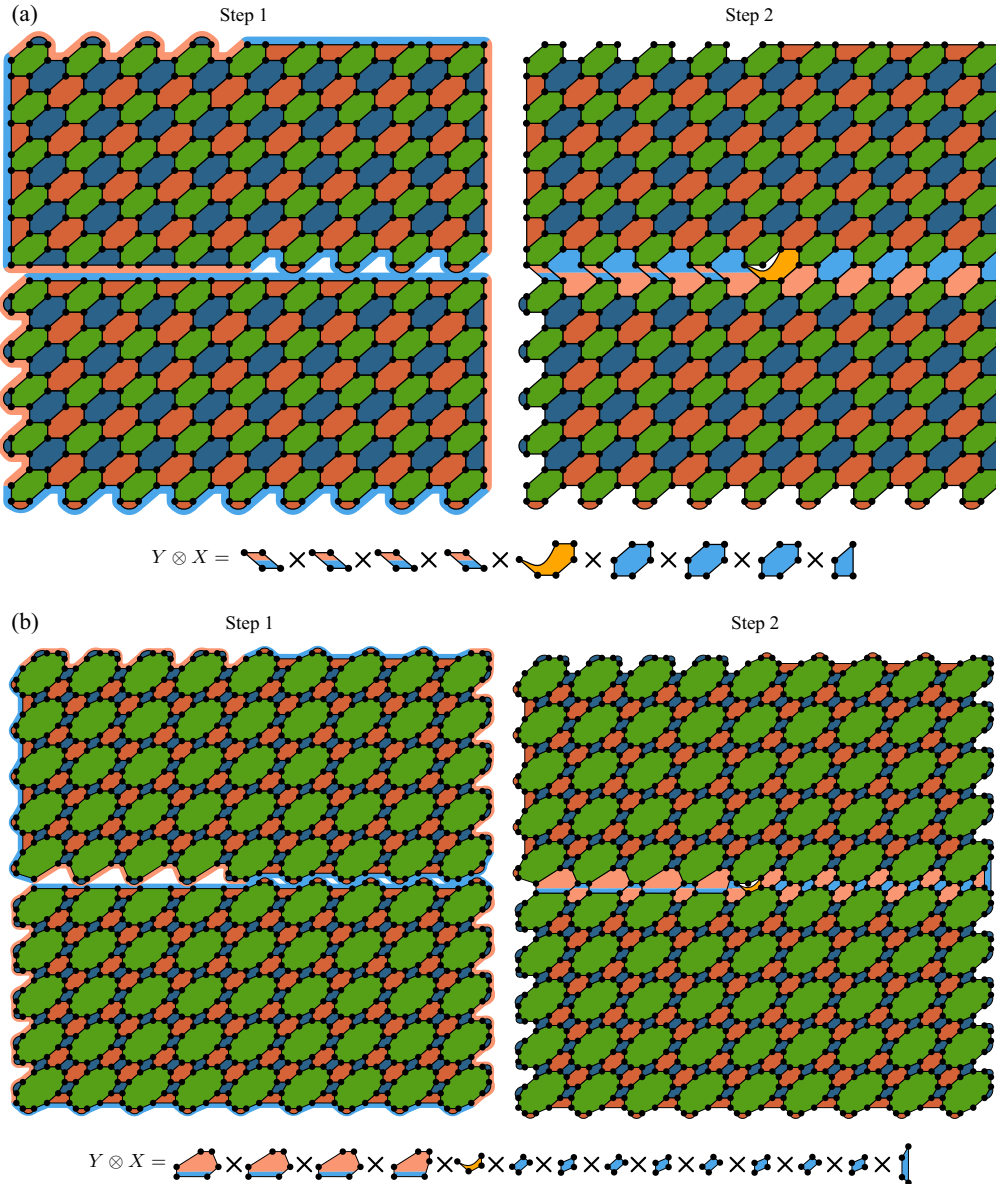


FIG. 11. Lattice surgery protocol for the measurement of $Y \otimes X$ between 6.6.6 (a) and 4.6.12 (b) Majorana surface code hexons and tetrons. The maximum weight of the operators during lattice surgery is 8 Majoranas for 6.6.6 codes and 6 Majoranas for 4.6.12 codes.

they only have two different boundaries each and therefore only two corners. The encoded qubit is in fact the physical tetron in the center. Notice that the product of the Majoranas highlighted in red would correspond to a logical X operator in an actual $d = 5$ code since it is a string connecting the two red boundaries that commutes with all stabilizers. In the stabilizer configuration in the top panel, this string also commutes with all stabilizers. Since the product of the Majoranas that are not located at the center tetron is fixed to be $+1$ by the stabilizers, the product of red Majoranas is equal to the X information encoded in the center tetron. Similarly, the product of blue Majoranas is equivalent to the Z information encoded in the center tetron, and would correspond to a blue-to-blue string of Majoranas in a $d = 5$ code. By switching the stabilizer configuration from the top configuration to the bottom configuration, the

state encoded in the physical tetron is converted into a logical tetron. As the highlighted logical operators commute with the stabilizers in both cases, they remain unchanged in the process. However, the measurement outcomes of the new stabilizers straddling the corner segments are random. The errors need to be corrected in such a way that the correction operation commutes with the highlighted logical operators. Note that since we start out with a $d = 1$ qubit, state injection is not fault tolerant, which further emphasizes the need for magic-state distillation.

The injection procedures for 6.6.6 and 4.6.12 Majorana surface codes are very similar. In the 6.6.6 case in Fig. 12(b), the logical information is initially encoded in 4 of the 6 Majoranas of the center hexon. Again, the highlighted Majorana strings commute with both stabilizer configurations and correspond

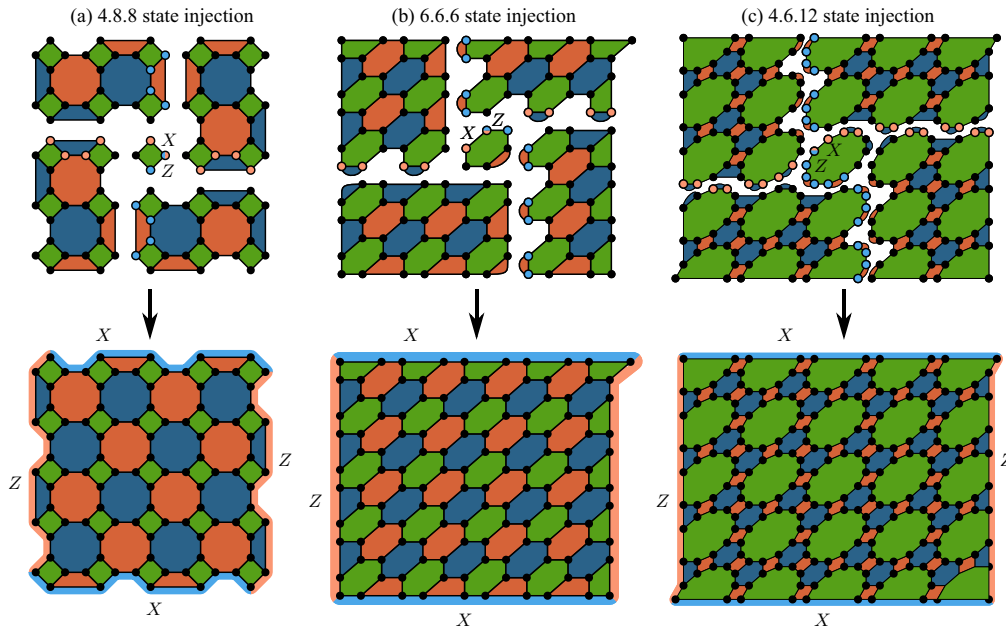


FIG. 12. State injection protocols to convert a state encoded in a physical tetron into a logical 4.8.8 (a), 6.6.6 (b), or 4.6.12 (c) tetron for the example of $d = 5$.

to the logical operators. For the 4.6.12 code in Fig. 12(c), the logical information is initially encoded in 4 of the 12 Majoranas of the center dodecon.

Thus, we have implemented all four operations necessary for universal quantum computation with tetrons and hexons with minimal-overhead Clifford gates. The Pauli product measurement protocol of Fig. 7 can be implemented with *logical* tetrons and hexons exactly the same way as was described in Sec. III by replacing the measurements of 4 Majoranas by (twist-based) lattice surgery.

V. MAJORANA COLOR CODES

In the previous section, we have shown that Majorana surface codes can be used to decrease the weight of the operators that need to be measured for fault-tolerant quantum computing. However, they come with a higher space overhead compared to bosonic surface codes. What about the opposite direction? How does one design topological codes with higher stabilizer weights but, in turn, lower space overhead?

The answer lies in color codes. Bosonic color codes [28,49] are known to feature higher stabilizer weights compared to bosonic surface codes, but they can encode qubits more compactly, using fewer physical qubits per logical qubit of a given code distance. Color codes are closely related to surface codes, as they can be obtained by concatenating surface codes with small nontopological codes [36]. In an alternative construction, color codes are obtained by folding surface codes in half [50]. Here, we adapt the concatenation scheme to Majorana surface codes, in order to obtain Majorana color codes.²

For bosonic codes, concatenation means that each physical qubit of a code is replaced by another logical qubit. For instance, a 4.8.8 bosonic color code can be obtained by concatenating a bosonic surface code with a $[[4,2,2]]$ code [36]. This means that two bosonic surface codes are stacked on top of each other, and pairs of stacked qubits are replaced by $[[4,2,2]]$ codes, as shown in Figs. 13(a) and 13(b). Similarly, we propose that Majorana color codes can be obtained by concatenating Majorana surface codes with $[[n_m, k, d_m]]_m$ codes, where $[[n_m, k, d_m]]_m$ labels Majorana fermion codes that use n Majoranas to encode k qubits with Majorana distance d_m .

We suggest to concatenate fermionic codes by stacking Majorana surface codes on top of each other and replacing stacked *building blocks* of the code by $[[n_m, k, d_m]]_m$ codes. The simplest example of a Majorana color code is a 4.8.8 Majorana surface code concatenated with a $[[6,2,2]]_m$ code, which is shown in Fig. 14. The $[[6,2,2]]_m$ code uses 6 Majoranas to encode two qubits with Majorana distance $d_m = 2$, which means that it is a hexon. In the concatenation procedure shown in Fig. 14 for a small segment of a 4.8.8 surface code, two Majorana surface codes are placed on top of each other. Next, each stack of two tetrons is replaced by a single hexon. Hexons encode twice as many qubits as tetrons, but use only 1.5 times as many Majoranas to do so. Thus, a logical tetron built from the Majorana color codes shown in Fig. 14 encodes two qubits as two layers of surface codes on top of each other, but uses only $3d^2$ Majoranas per qubit with distance d , while the stabilizer weight remains the same. Note that even though we draw Majorana color codes as three-dimensional stacks, they can also be implemented in 2D architectures. In particular, the 4.8.8 $([[6,2,2]]_m)$ Majorana color code can be implemented in a square lattice of hexons, but, in contrast to Majorana surface codes, the stabilizers now overlap, as we show in Appendix C 1.

More generally, we define Majorana color codes as any number of Majorana surface code layers obtained by

²Note that in Ref. [14], the term *Majorana color code* was used to refer to Majorana surface codes.

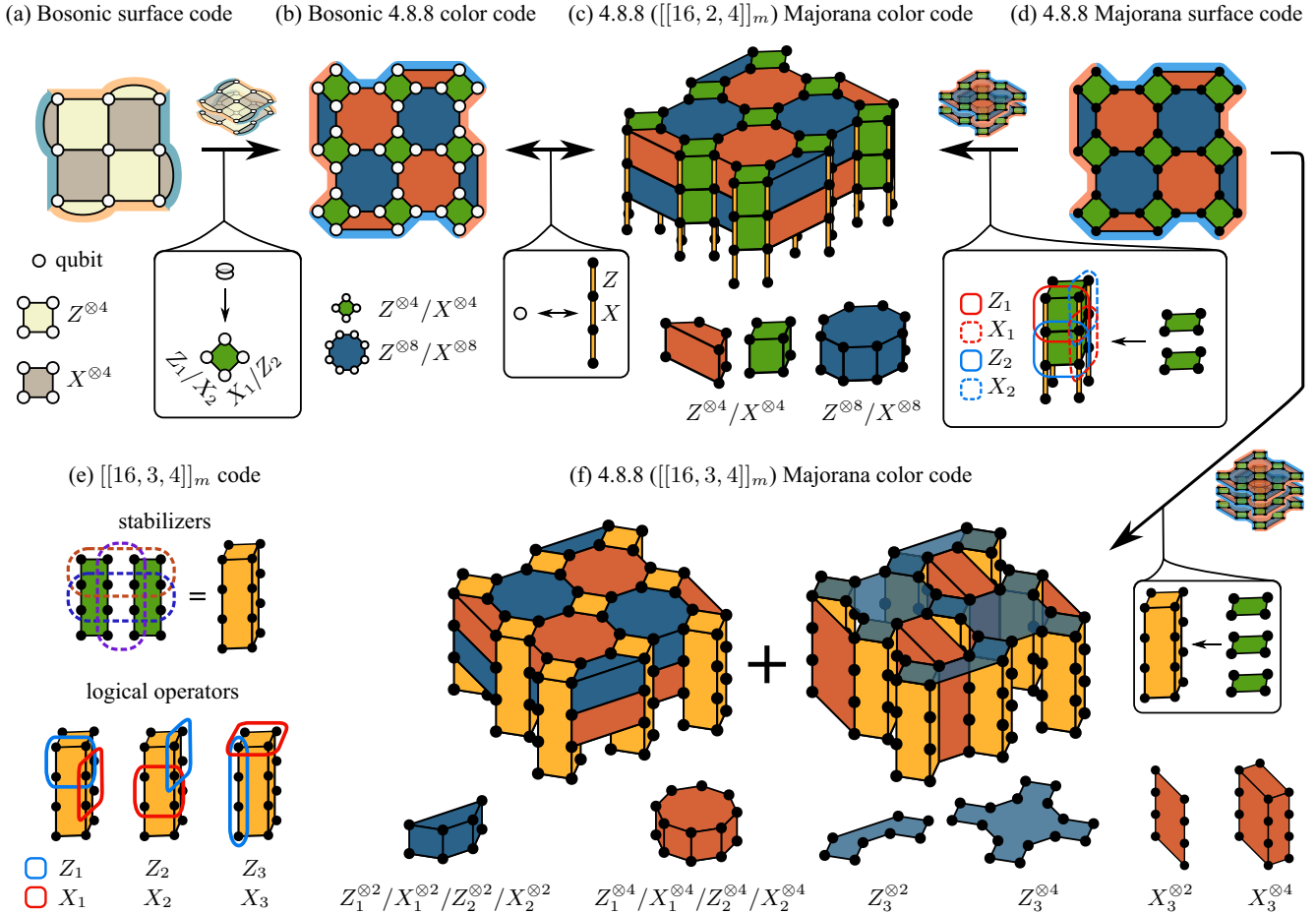
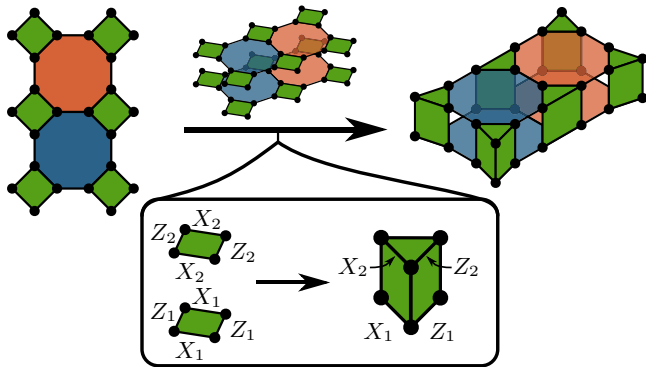


FIG. 13. A bosonic surface code (a) concatenated with a $[[4,2,2]]$ code yields a bosonic 4.8.8 code (b). This can also be drawn as a Majorana fermion code (c) by replacing each qubit with a tetron (yellow lines). This Majorana color code can also be obtained by replacing each tetron of the 4.8.8 Majorana surface code with a $[[16,2,4]]_m$ code. (e) Stabilizers and logical operators of the $[[16,3,4]]_m$ code presented in Ref. [24]. (f) Order-3 Majorana color code obtained by concatenating a 4.8.8 Majorana surface code with the $[[16,3,4]]_m$ code. Each yellow 16-Majorana box corresponds to a $[[16,3,4]]_m$ code. Since the stabilizers overlap, they are shown in two separate figures.

concatenating Majorana surface codes with any other code. We refer to a code corresponding to n layers of surface codes as an order- n code. Stacking k 4.8.8 Majorana surface codes

and replacing each stack of k tetrons with an $[[n_m, k, d_m]]_m$ code yields an order- k Majorana color code. For instance, the previously discussed 4.8.8 $[[6,2,2]]_m$ Majorana color code is an order-2 Majorana color code. Majorana surface codes are order-1 codes. One way to obtain an order-3 code is to place three 4.8.8 Majorana surface codes on top of each other, and replace each stack of three tetrons with an octon (an $[[8,3,2]]_m$ code). Since one of the logical operators of the octon is a 4-Majorana operator, the maximum stabilizer weight increases to 10 Majoranas, as we discuss in Appendix C 2.



Bosonic color codes in their usual definition [28] are order-2 codes, as they correspond to two surface code layers. One such code is the previously mentioned bosonic 4.8.8 color code, which is obtained by replacing each qubit of a bosonic surface code [Fig. 13(a)] with a $[[4,2,2]]$ code. This is a code that uses four qubits and has two stabilizers $Z \otimes Z \otimes Z \otimes Z$ and $X \otimes X \otimes X \otimes X$. It encodes two logical qubits with logical operators $Z_1 = Z \otimes Z \otimes \mathbb{1} \otimes \mathbb{1}$, $X_1 = \mathbb{1} \otimes X \otimes X \otimes \mathbb{1}$ and $Z_2 = \mathbb{1} \otimes Z \otimes Z \otimes \mathbb{1}$, $X_2 = X \otimes X \otimes \mathbb{1} \otimes \mathbb{1}$. Thus, a bosonic 4.8.8 color code tetron [see Fig. 13(b)] is equivalent to two surface codes on top of each other. It can be converted to a Majorana fermion code by replacing each qubit with

FIG. 14. Scheme to obtain a 4.8.8 $[[6,2,2]]_m$ Majorana color code by concatenating the 4.8.8 Majorana surface code with a $[[6,2,2]]_m$ code, i.e., by stacking two surface codes and replacing each pair of tetrons with a hexon.

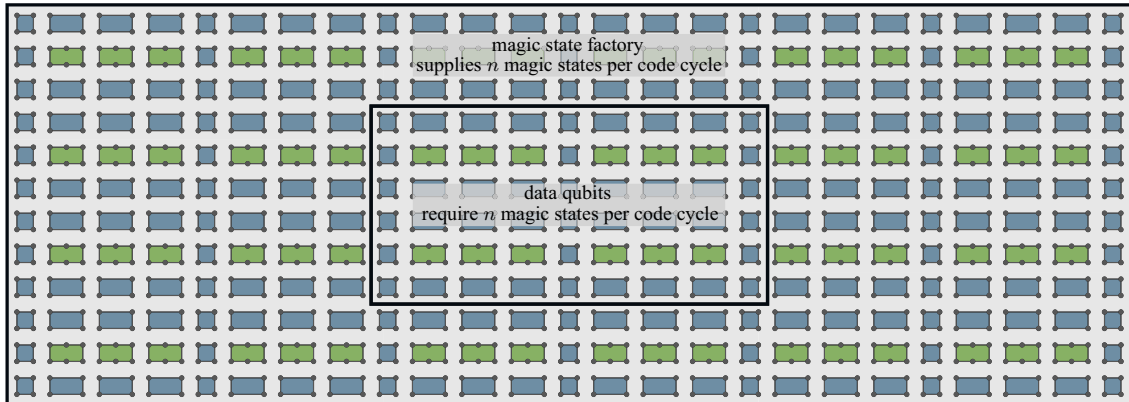


FIG. 15. Scheme for quantum computation in an array of tetrons and hexons, where the hexons are used to encode qubits, and tetrons are used as ancillas for Pauli product measurements. One part of the array is used to encode the data qubits for quantum computation, whereas the rest of the quantum computer is used to distill magic states.

a tetron. In Fig. 13(c), these tetron building blocks are the yellow strings of 4 Majoranas. Another way of obtaining this Majorana fermion code is by stacking two 4.8.8 Majorana surface codes and replacing pairs of tetrons with $[[16,2,4]]_m$ codes, which is the Majorana representation of the bosonic $[[4,2,2]]$ code. The bosonic 4.8.8 color code uses four times as many physical qubits (or Majoranas) to encode twice as many logical qubits with twice the code distance. Thus, the Majorana overhead of bosonic 4.8.8 color codes is $2d^2$ per qubit, which is half the overhead of bosonic surface codes. However, the stabilizer weight doubles from 8 Majoranas to 16 Majoranas.

The $[[16,2,4]]_m$ code used to obtain the bosonic 4.8.8 color code by concatenation is a bosonic code in the sense that its building blocks are tetrons. With Majoranas, there is also the possibility to use Majorana fermion codes for concatenation. These codes can encode information more compactly. In Ref. [24], a collection of small Majorana fermion codes is presented. The smallest $d_m = 4$ Majorana fermion code is the $[[16,3,4]]_m$ code shown in Fig. 13(e), which uses 16 Majoranas to encode three instead of just two logical qubits with a Majorana distance of $d_m = 4$. It is based on two octons with three additional 8-Majorana operators as stabilizers. All of its logical operators are weight-4-Majorana operators. The order-3 Majorana color code obtained by concatenating the 4.8.8 surface code with the $[[16,3,4]]_m$ code is shown in Fig. 13(f). It essentially corresponds to three layers of Majorana surface codes and has a reduced Majorana overhead of $\frac{4}{3}d^2$ with octons as building blocks, while still featuring a maximum stabilizer weight of 16 Majoranas.

In general, concatenating a bosonic surface code with an $[[n,k,d_b]]$ bosonic code increases the number of physical qubits by a factor of n , the number of logical qubits by a factor of k , and the code distance by a factor of d_b . Thus, the number of physical qubits per logical qubit goes from d^2 to $\frac{n}{kd_b}d^2$. The maximum stabilizer weight of the code may increase from 4 to $4w_{\max}$, where w_{\max} is the maximum weight of the logical operators of the $[[n,k,d_b]]$ code. Similarly, a 4.8.8 Majorana surface code concatenated with an $[[n_m,k,d_m]]_m$ code decreases the Majorana overhead from $4d^2$ to $\frac{4n_m}{kd_m^2}d^2$ Majoranas for a logical qubit with code distance d .

The second smallest $d_m = 4$ Majorana fermion code is the $[[20,4,4]]_m$ code. Its stabilizers and logical operators are shown in Appendix C3. The code is based on two decons (10 Majoranas in a fixed-parity sector) and four 10-Majorana stabilizers, and encodes four qubits whose logical operators all have a weight of 4 Majoranas. Thus, the order-4 Majorana color code obtained by concatenating a 4.8.8 surface code with a $[[20,4,4]]_m$ has a Majorana overhead of $1.25d^2$ while still only featuring a maximum stabilizer weight of 16 Majoranas. Since some of the logical operators of the third-smallest $d_m = 4$ code, the $[[24,6,4]]_m$ code, have a weight of 6 Majoranas, the stabilizer weight of a Majorana code obtained by concatenating this code with a 4.8.8 Majorana surface code would exceed 16 Majoranas. In general, Majorana surface codes can be concatenated with arbitrary Majorana fermion codes to obtain codes with a smaller Majorana overhead, while increasingly sacrificing stabilizer weight and locality of the code.

Majorana color codes can be used for quantum computation the same way as Majorana surface codes, i.e., by encoding tetrons and hexons, and performing lattice surgery. In fact, the stabilizer weight and space overhead of the lattice surgery protocol can be reduced by using surface-to-color code lattice surgery [20], such that the ancilla tetron is not a color code, but a surface code. For the previously discussed codes with 16-Majorana stabilizers, surface-to-color code lattice surgery has a maximum stabilizer weight of 12 Majoranas. We show an example of this in Appendix C4.

In summary, we have described Majorana color codes, which are multiple layers of Majorana surface codes obtained by concatenation. The advantage of Majorana color codes compared to bosonic color codes is that they can encode logical qubits more compactly with the same stabilizer weight. In particular, the Majorana color code obtained by concatenating the 4.8.8 Majorana surface code with the $[[20,4,4]]_m$ code has a Majorana overhead of $1.25d^2$, compared to the Majorana overhead of $2d^2$ of the bosonic 4.8.8 color code.

VI. CONCLUSION

We have described a unified framework for fault-tolerant quantum computation with Majorana-based qubits. A full

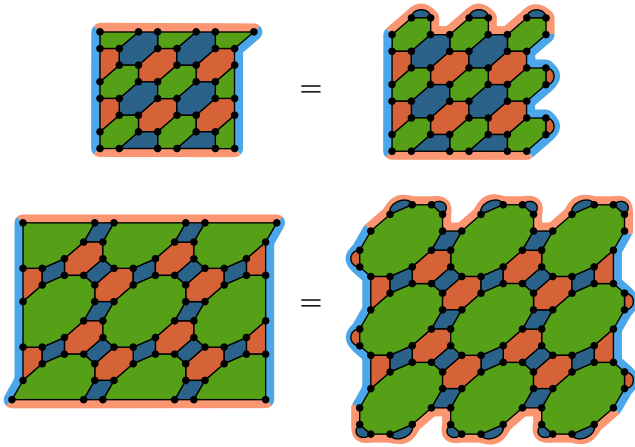


FIG. 16. 6.6.6 and 4.6.12 surface code tetrons in a square lattice of hexons and dodecons, respectively.

quantum computer could look like the array of tetrons and hexons shown in Fig. 15. Hexons are used to encode qubits, while tetrons are ancillas that are used during the Pauli product measurement protocol. One part of the quantum computer

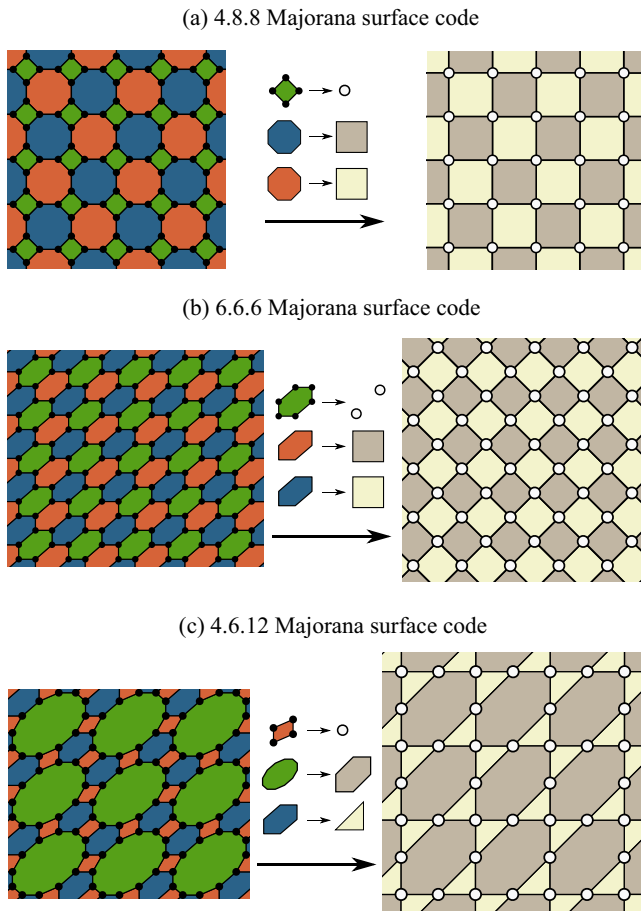


FIG. 17. One choice of mapping of 4.8.8, 6.6.6, and 4.6.12 Majorana surface codes onto bosonic surface codes. For bosonic surface codes, the dark plaquettes are products of X operators, and the light plaquettes are products of Z operators.

$$|\psi\rangle \text{---} T \text{---} = |\psi\rangle \begin{array}{c} \begin{array}{|c|} \hline Z \\ \hline \end{array} \\ \begin{array}{|c|} \hline X \\ \hline \end{array} \end{array} \begin{array}{c} m_1 \\ m_2 \end{array} \begin{array}{c} S^{m_1} \\ X^{m_2} \end{array} \begin{array}{c} Z^{m_2} \\ \end{array}$$

FIG. 18. A T gate on a qubit $|\psi\rangle$ is equivalent to a $Z \otimes Z$ measurement between $|\psi\rangle$ and a magic state $|m\rangle$ with outcome m_1 , which results in a corrective S^{m_1} operation on $|\psi\rangle$. In order to disentangle $|m\rangle$ from the qubit, it is measured in the X basis, prompting a Z^{m_2} Pauli correction on $|\psi\rangle$.

consists of the data qubits that are used for quantum computation, which requires a certain number of magic states n per code cycle as a resource for T gates, where n depends on the given quantum computation. In order for magic-state distillation not to be a bottleneck of the quantum computer, the resources devoted to magic-state distillation need to be large enough to supply n magic states per code cycle. The main operation in addition to the preparation of faulty magic states is the measurement of Pauli product operators according to the protocol in Fig. 7. Taking full advantage of the Gottesman-Knill theorem, this enables the classical tracking of all Clifford gates, which means that these gates require zero-time overhead.

In principle, the scheme with $d = 1$ tetrons and hexons shown in Fig. 15 can be implemented in a nanowire array in the spirit of Ref. [33], as shown in Appendix B 1. However, this only allows for quantum computation on timescales of the order of the coherence times of the physical tetrons and hexons. For fault-tolerant quantum computation, each tetron and hexon needs to be replaced by a *logical* tetron and hexon with an appropriate code distance, and 4-Majorana parity measurements need to be replaced by lattice surgery.

There are a wide variety of codes that one can choose for this purpose, a collection of which is shown in Table I. Surface codes only require the measurement of low-weight stabilizers, but feature a high-space overhead. Majorana surface codes feature lower-weight stabilizers compared to bosonic surface codes. In particular, 4.8.8 Majorana surface codes have the same bulk stabilizer weights as bosonic surface codes, but a lower stabilizer weight for lattice surgery of 8 Majoranas compared to 10 Majoranas. 6.6.6 and 4.6.12 Majorana surface codes reduce the bulk stabilizer weights to an average

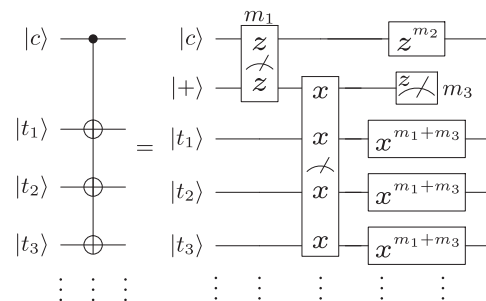


FIG. 19. A multitarget CNOT gate between a control qubit $|c\rangle$ and multiple target qubits $|t_i\rangle$ is equivalent to a $Z \otimes Z$ measurement between $|c\rangle$ and an ancilla initialized in the $|+\rangle$ state, followed by a $X^{\otimes n+1}$ measurement between the ancilla and the n target qubits. Finally, the ancilla is measured in the Z basis. The final Pauli corrections depend on the measurement results.

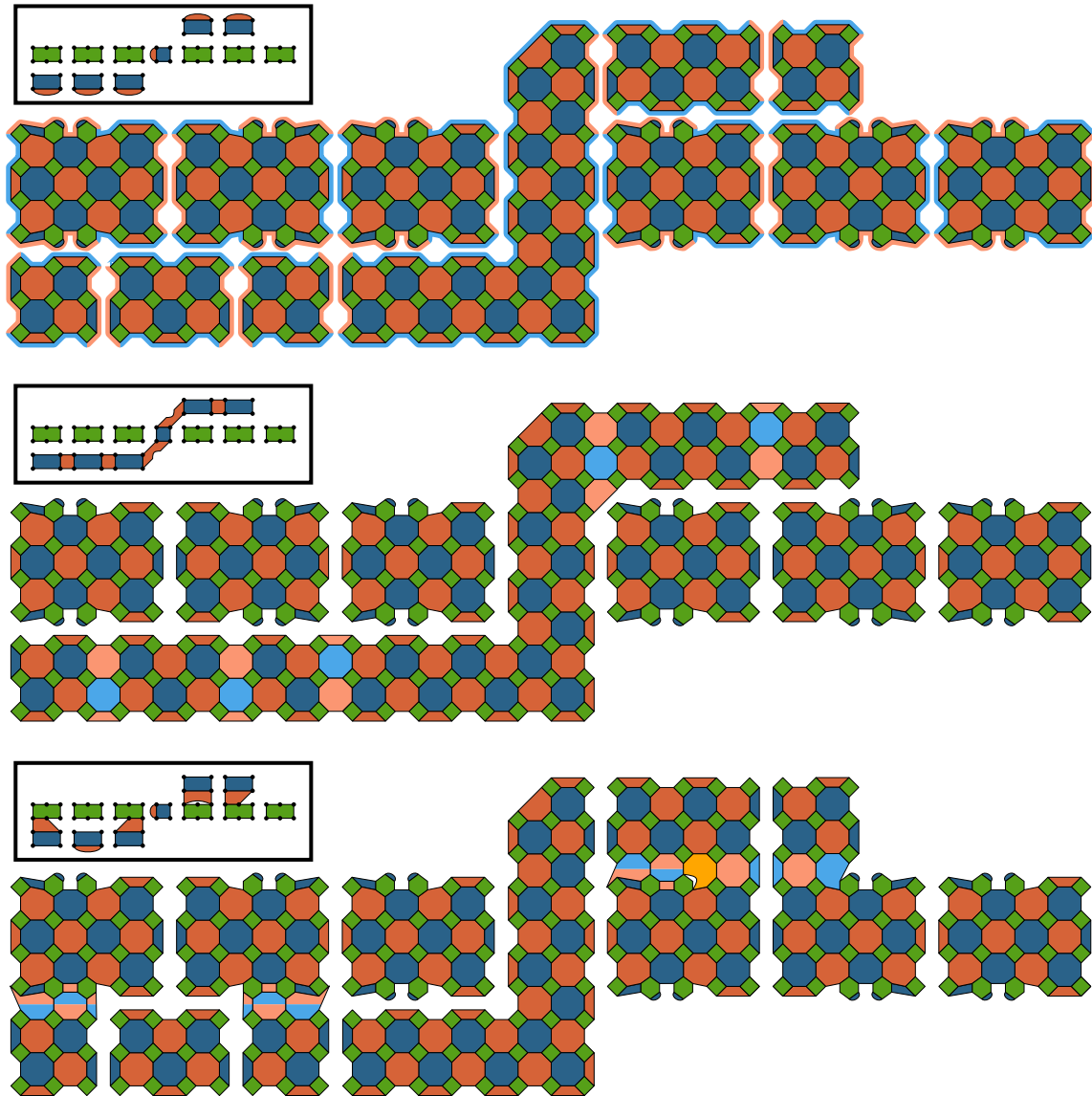


FIG. 20. Pauli product measurement protocol from Fig. 7 realized with $d = 3$ 4.8.8 Majorana surface codes. Only steps 1, 2, and 3 are shown.

of 6 and 5 Majoranas, respectively, but at the same time increase the Majorana overhead by a factor of 1.5 and 3, respectively.

Color codes can reduce the space overhead at the price of higher-weight stabilizers. Majorana color codes encode qubits more compactly compared to bosonic color codes. The 4.8.8 $([[6,2,2]]_m)$ Majorana color code has the same stabilizer weights as a bosonic surface code, but features a lower Majorana overhead. Similarly, the 4.8.8 $([[20,4,4]]_m)$ Majorana color code has the same stabilizer weights as the bosonic 4.8.8 color code, but a lower Majorana overhead by a factor of $\frac{5}{8}$.

Our entire scheme can also be realized with nontopological (e.g., superconducting) qubits, but is then limited to the use of bosonic codes. From the perspective of Majorana fermion codes, nontopological qubits can only implement logical tetrons and hexons with physical tetrons as building blocks,

whereas Majorana-based qubits can also implement codes with other building blocks such as physical hexons, octons, decons, or dodecons.

There are many questions that remain unanswered. Even though our protocols allow the classical tracking of CNOT gates, it remains uncertain whether tracking is always advantageous compared to actually executing these gates. After all, if a layer of Clifford gates is followed by a layer of T gates, all of these T gates can be executed simultaneously. However, if CNOTs are tracked, these T gates require spatially overlapping Pauli product measurements. While our measurement protocol shows how arbitrary Pauli products can be measured, it is unclear how multiple (commuting) Pauli products can be read out simultaneously. If it turns out that it is more advantageous to execute CNOT gates explicitly, our Pauli product measurement protocol can be straightforwardly used for multitarget CNOT gates using the protocol in Appendix A3.

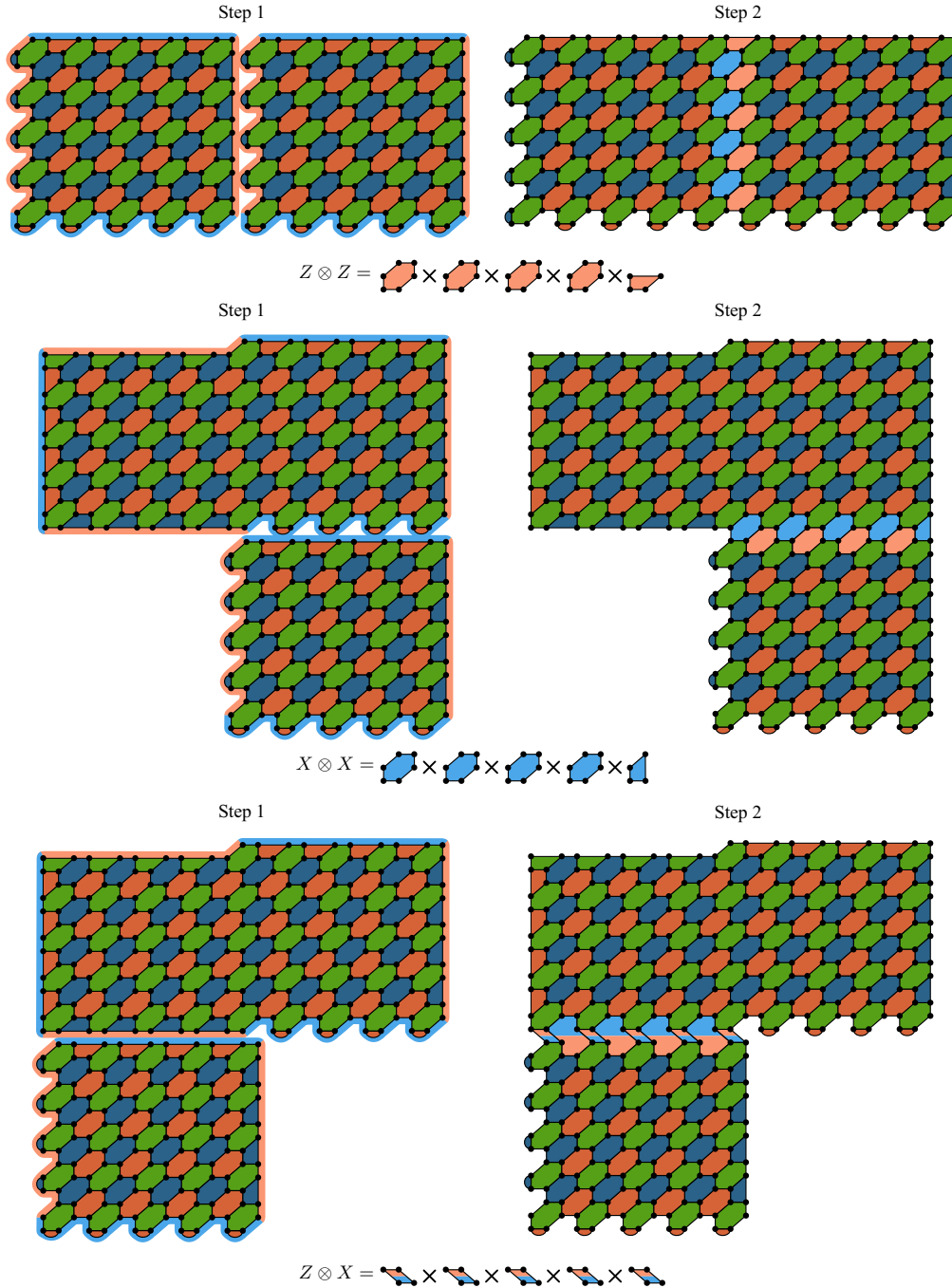


FIG. 21. Stabilizer configuration for $Z \otimes Z$ measurements between two 6.6.6 Majorana surface code tetrons, and for $X \otimes X$ and $Z \otimes X$ measurements between a 6.6.6 hexon and a 6.6.6 tetron.

Moreover, while bosonic surface code decoders can be used to decode Majorana surface codes, one could also devise decoding schemes that are tailored towards Majorana fermion codes. In particular, in a setting where parity-fixing constraints can be violated and stabilizers of all colors are measured, a decoder needs to be able to match three types of anyons, as opposed to just two. Furthermore, one could adapt the scheme of Majorana-based fermionic quantum computation [51] to the fault-tolerant setting by replacing arrays of Majoranas with arrays of twist defects in Majorana surface codes. Based on

twists in Majorana surface codes, it would also be interesting to see whether it is possible to come up with a Majorana version of the twist-based triangle code presented in Ref. [52]. Also, we did not investigate the performance of Majorana fermion codes, i.e., the logical error rate under the assumption of a realistic error model. While it is known that the bit-flip error thresholds of bosonic surface and color codes without measurement errors are the same [53,54], the error thresholds of codes corresponding to an arbitrary number of surface code layers are unknown. We hope that the framework presented

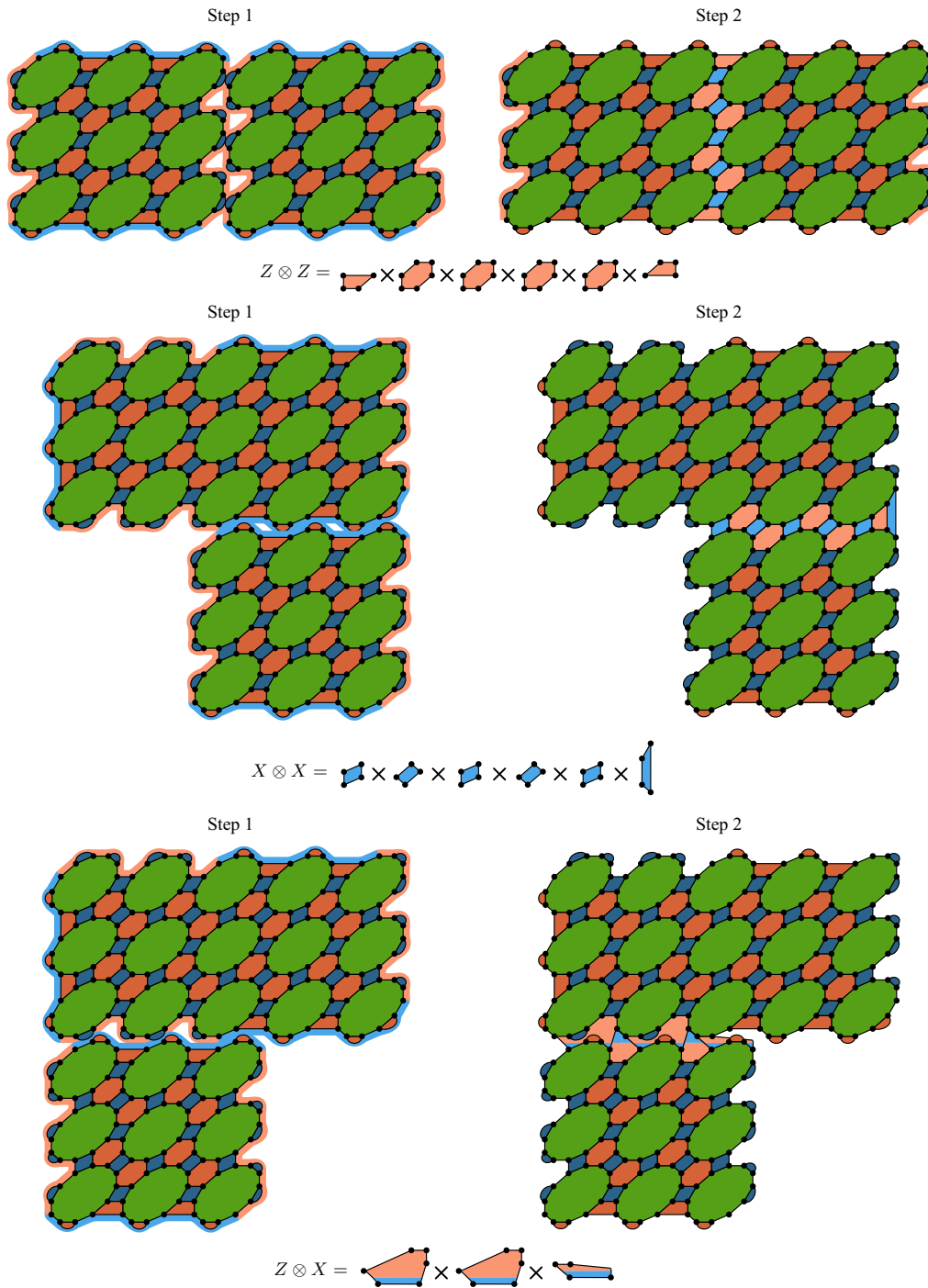


FIG. 22. Stabilizer configuration for $Z \otimes Z$ measurements between two 4.6.12 Majorana surface code tetrons, and for $X \otimes X$ and $Z \otimes X$ measurements between a 4.6.12 hexon and a 4.6.12 tetron.

in this work will prove useful for Majorana-based quantum computation.

ACKNOWLEDGMENTS

The authors would like to thank M. Kesselring, J. Eisert, T. Yoder, M. Beverland, B. Brown, and A. Kubica for insightful discussions. This work has been financially supported by the Deutsche Forschungsgemeinschaft (Bonn) within the network CRC TR 183.

APPENDIX A: MAJORANA SURFACE CODES

1. 6.6.6 and 4.6.12 Majorana surface codes in square lattices of hexons and dodecons

Some of the logical surface code tetrons and hexons in Figs. 3 and 4 feature smaller building blocks (green stabilizers) at the boundary compared to the building blocks in the bulk. In particular, some of the green stabilizers of 6.6.6 surface codes are tetrons at the boundary, whereas they are hexons in the bulk.

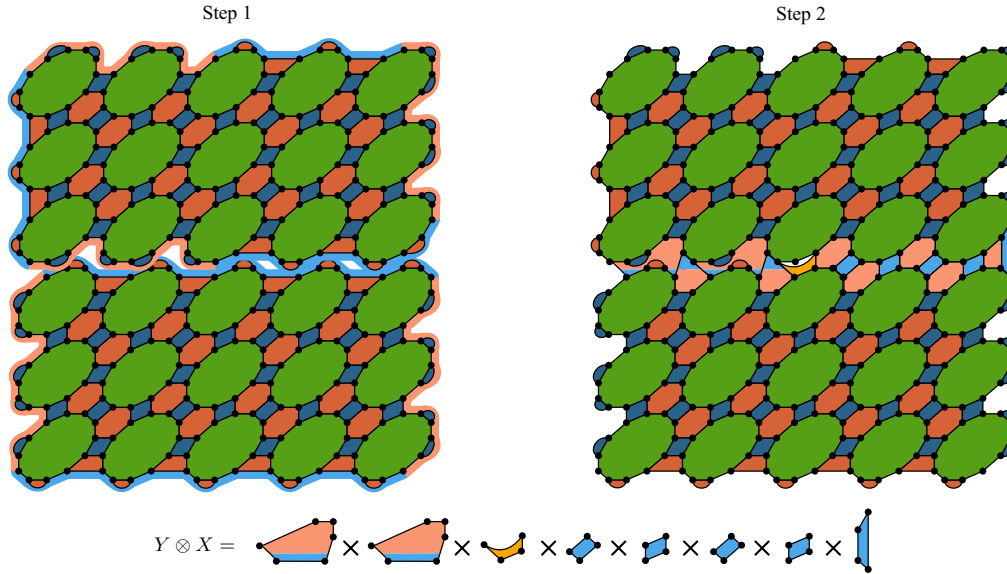


FIG. 23. Stabilizer configuration for $Y \otimes Z$ measurements between a 4.6.12 Majorana hexon and tetron.

Similarly, 4.6.12 surface codes feature hexons, octons, and decons at the boundary, as opposed to dodecons. Still, all of these codes can be implemented on a square lattice of building blocks (green stabilizers), where each building block is the same. The equivalence in Fig. 16 shows that 6.6.6 Majorana surface codes can be implemented on a square lattice of hexons, and 4.6.12 codes can be implemented on a square lattice of dodecons. The two-qubit plaquettes at the boundary fix the parity of some of the Majoranas of the hexons and dodecons, such that they effectively become tetrons, hexons, octons, or decons.

2. Mapping Majorana surface codes onto bosonic surface codes

While every bosonic code can be uniquely mapped onto a Majorana fermion code by replacing each qubit with a tetron, the reverse mapping is not unique. For Majorana surface codes, one prescription to map them onto a bosonic code is to interpret each stabilizer of a certain color of the three-colorable tiling as a number of qubits. For instance, for the 4.8.8 tiling, one has the choice of either interpreting the 8-Majorana stabilizers of one color as three qubits encoded in an octon, or of interpreting all

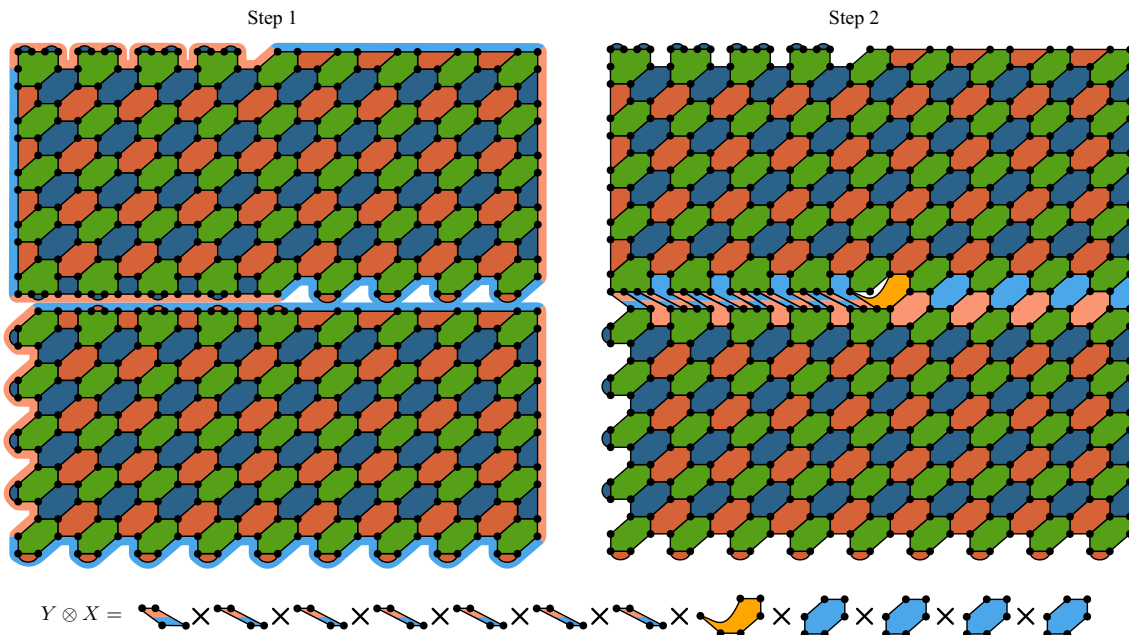


FIG. 24. Stabilizer configuration for $Y \otimes Z$ measurements between a 6.6.6 Majorana hexon and tetron. Some of the physical hexons located at the boundary have been replaced by octons. As a consequence, the maximum stabilizer weight reduces to 6 Majoranas.

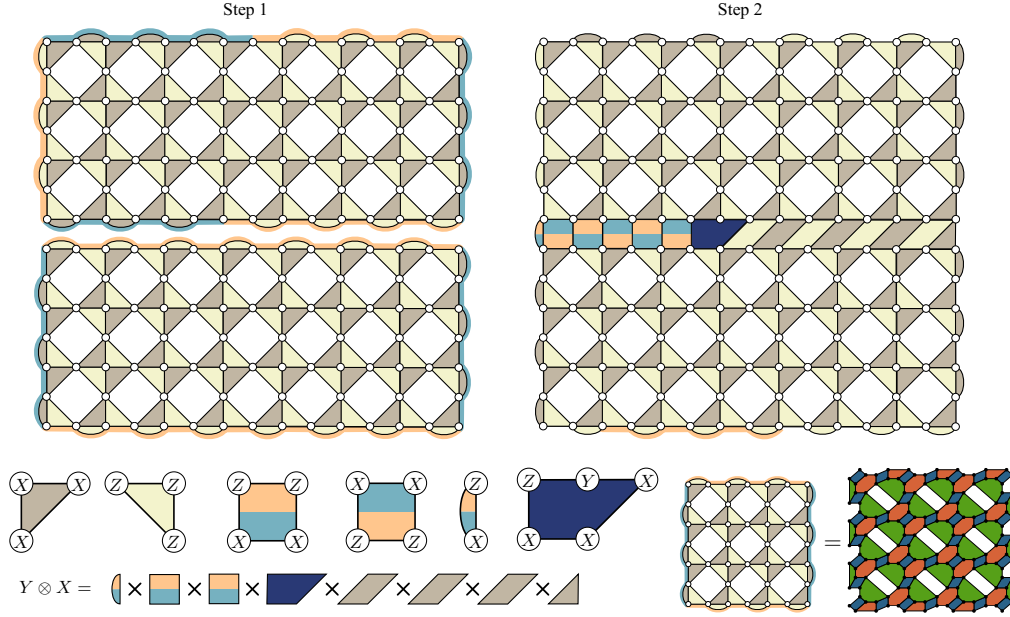


FIG. 25. Twist-based lattice surgery between a logical hexon and a logical tetron encoded in a $d = 3$ bosonic subsystem surface code in order to measure $Y \otimes Z$. The Majorana fermion code obtained by replacing each qubit of a bosonic subsystem surface code tetron with 4 Majoranas in a fixed-parity sector resembles a 4.6.12 Majorana surface code.

4-Majorana stabilizers as tetrons. The latter mapping precisely yields the bosonic surface code on a square lattice, as shown in Fig. 17(a). Note that in this mapping, neighboring tetrons have different orientations.

For the 6.6.6 Majorana surface code in Fig. 17(b), we interpret each green stabilizer as a hexon, where one qubit is encoded in the bottom left three Majoranas, and the other qubit in the top right three Majoranas. Thus, we replace each green hexon with two qubits, and obtain a bosonic surface code on a rotated square lattice. For the 4.6.12 Majorana surface codes there are again several possibilities. In Fig. 17(c), we show a mapping where each 4-Majorana stabilizer is replaced by a tetron, which yields a bosonic surface code on a kagome lattice.

3. State injection and CNOT gates with Pauli product measurements

In a quantum computing architecture where all Clifford gates are tracked and only Pauli product measurements are explicitly performed, the state injection circuit from Fig. 5(a) is replaced by Fig. 18. In particular, the tracked CNOT gate maps the Z measurement of the magic state onto a $Z \otimes Z$ measurement of $|\psi\rangle \otimes |m\rangle$. However, this leaves the qubits $|\psi\rangle$ and $|m\rangle$ in an entangled state, which means that $|m\rangle$ cannot be discarded right away after the state injection. With $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and $|m\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)$, the initial state before any measurement is

$$|\Psi\rangle = \frac{1}{\sqrt{2}}(\alpha|00\rangle + \alpha e^{i\pi/4}|01\rangle + \beta|10\rangle + \beta e^{i\pi/4}|11\rangle). \quad (\text{A1})$$

After the $Z \otimes Z$ measurement, there are two possible outcomes:

$$|\Psi_0\rangle = \alpha|00\rangle + \beta e^{i\pi/4}|11\rangle \quad (\text{A2})$$

for outcome $m_1 = 0$, and

$$|\Psi_1\rangle = \alpha e^{i\pi/4}|01\rangle + \beta|10\rangle \approx \alpha|01\rangle + \beta e^{-i\pi/4}|10\rangle \quad (\text{A3})$$

for outcome $m_1 = 1$, where “ \approx ” is an equality up to a global phase. After an S^{m_1} correction, $|\Psi_0\rangle$ remains unchanged, whereas $|\Psi_1\rangle$ is transformed to

$$S|\Psi_1\rangle = \alpha|01\rangle + \beta e^{i\pi/4}|10\rangle. \quad (\text{A4})$$

Both $|\Psi_0\rangle$ and $S|\Psi_1\rangle$ are entangled states. If one wants to discard the second qubit in order to continue to use it for magic-state distillation, it first needs to be disentangled from the rest of the system. This can be done by measuring the magic state in the X basis with outcome m_2 , followed by a Z^{m_2} Pauli correction on the qubit $|\psi\rangle$. For $|\Psi_0\rangle$, an X measurement leaves the state in

$$|\Psi_{0,0}\rangle = (\alpha|0\rangle + \beta e^{i\pi/4}|1\rangle) \otimes |+\rangle \quad (\text{A5})$$

for outcome $m_2 = 0$, and

$$|\Psi_{0,1}\rangle = (\alpha|0\rangle - \beta e^{i\pi/4}|1\rangle) \otimes |-\rangle \quad (\text{A6})$$

for outcome $m_2 = 1$. A Z^{m_2} correction leaves $|\Psi_{0,0}\rangle$ unchanged and maps $|\Psi_{0,1}\rangle$ to

$$Z|\Psi_{0,1}\rangle = (\alpha|0\rangle + \beta e^{i\pi/4}|1\rangle) \otimes |-\rangle. \quad (\text{A7})$$

Similarly, $S|\Psi_1\rangle$ after an X measurement becomes $|\Psi_{0,0}\rangle$ for outcome $m_2 = 0$, and $-|\Psi_{0,1}\rangle \approx |\Psi_{0,1}\rangle$ for outcome $m_2 = 1$, which after a Z correction becomes $Z|\Psi_{0,1}\rangle$. For both $|\Psi_{0,0}\rangle$ and $Z|\Psi_{0,1}\rangle$, the two qubits are disentangled, and the first qubit is in the state $T|\psi\rangle = |0\rangle + e^{i\pi/4}|1\rangle$, which is the desired outcome for state injection. We stress that neither the S^{m_1} nor the Z^{m_2} correction need to be performed explicitly, but can be tracked by a classical computer.

We also note that in a quantum computer that only implements Pauli product measurements, CNOT gates can still

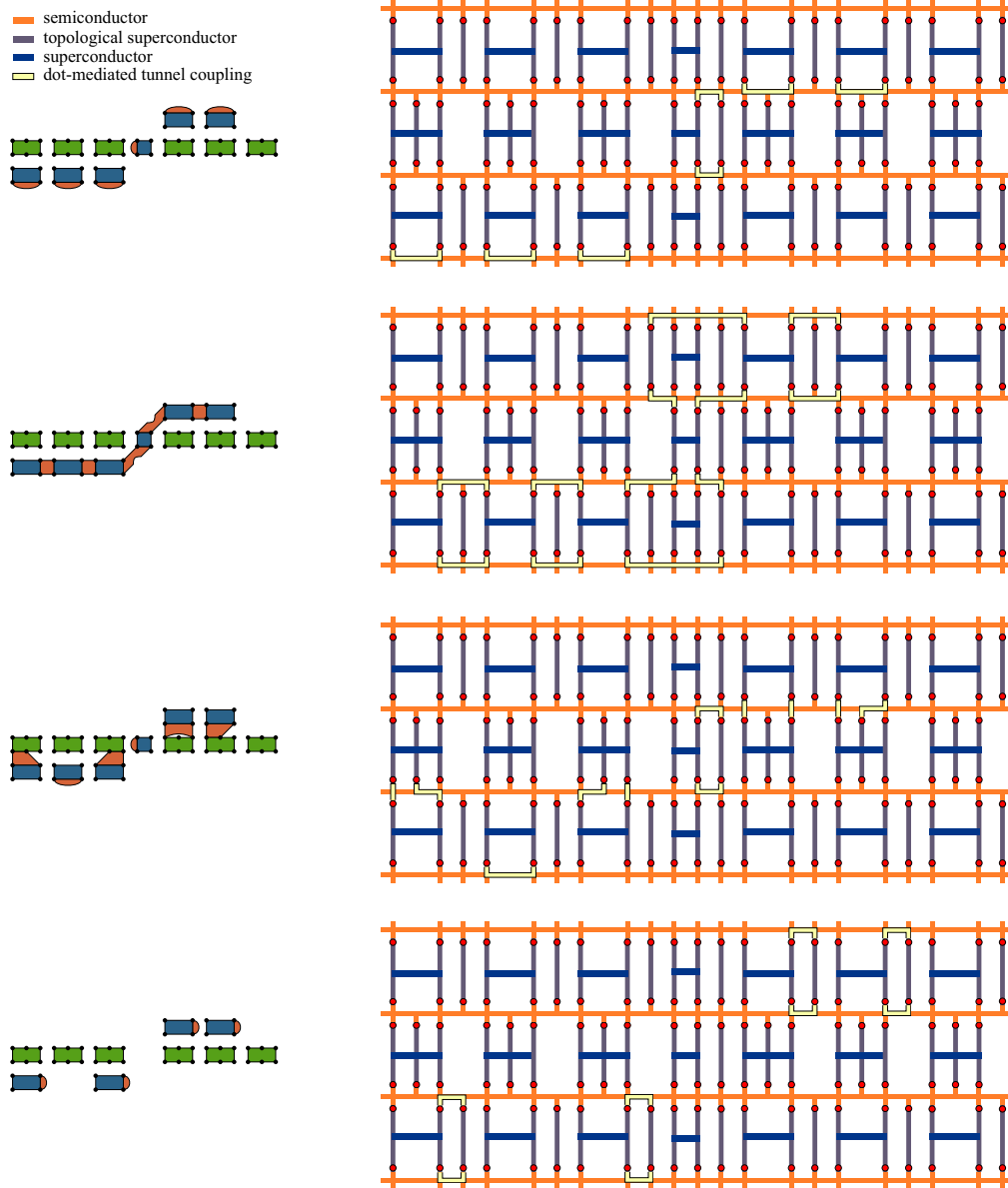


FIG. 26. Tunnel-coupling configurations for the Pauli product measurements protocol from Fig. 7 realized with the wire-based tetrons and hexons proposed in Ref. [33].

be performed explicitly. To this end, one can use the circuit identity for multitarget CNOT gates [19,55] shown in Fig. 19. Here, a CNOT gate between a control qubit $|c\rangle$ and n target qubit $|t_i\rangle$ is equivalent to three Pauli product measurements with Pauli corrections that depend on the measurement results.

4. Pauli product measurement protocol with 4.8.8 Majorana surface codes

In Fig. 20, we show how the example of a Pauli product measurement in Fig. 7 could be realized using 4.8.8 Majorana surface codes with $d = 3$. While the lattice surgery steps are shown, the X and Z measurements of logical tetrons are done via the measurement of all 2-Majorana operators corresponding to red and blue edges, respectively.

5. Lattice surgery protocols for 6.6.6 and 4.6.12 Majorana surface codes

In Fig. 21, we show the stabilizer configurations for the lattice surgery operations to measure $Z \otimes Z$ between two 6.6.6 Majorana surface code tetrons, and $X \otimes X$ and $Z \otimes X$ between hexons and tetrons. The same lattice surgery operations are shown for 4.6.12 Majorana surface codes in Fig. 22. Since Fig. 11(b) shows the $Y \otimes Z$ lattice surgery for 4.6.12 codes with decons along the boundary, we show the same protocol in Fig. 23, but on a square lattice of dodecons.

6. Weight-6 lattice surgery protocol for 6.6.6 Majorana surface codes

Figure 24 shows the lattice surgery protocol for a $Y \otimes X$ measurement between a 6.6.6 Majorana surface code hexon

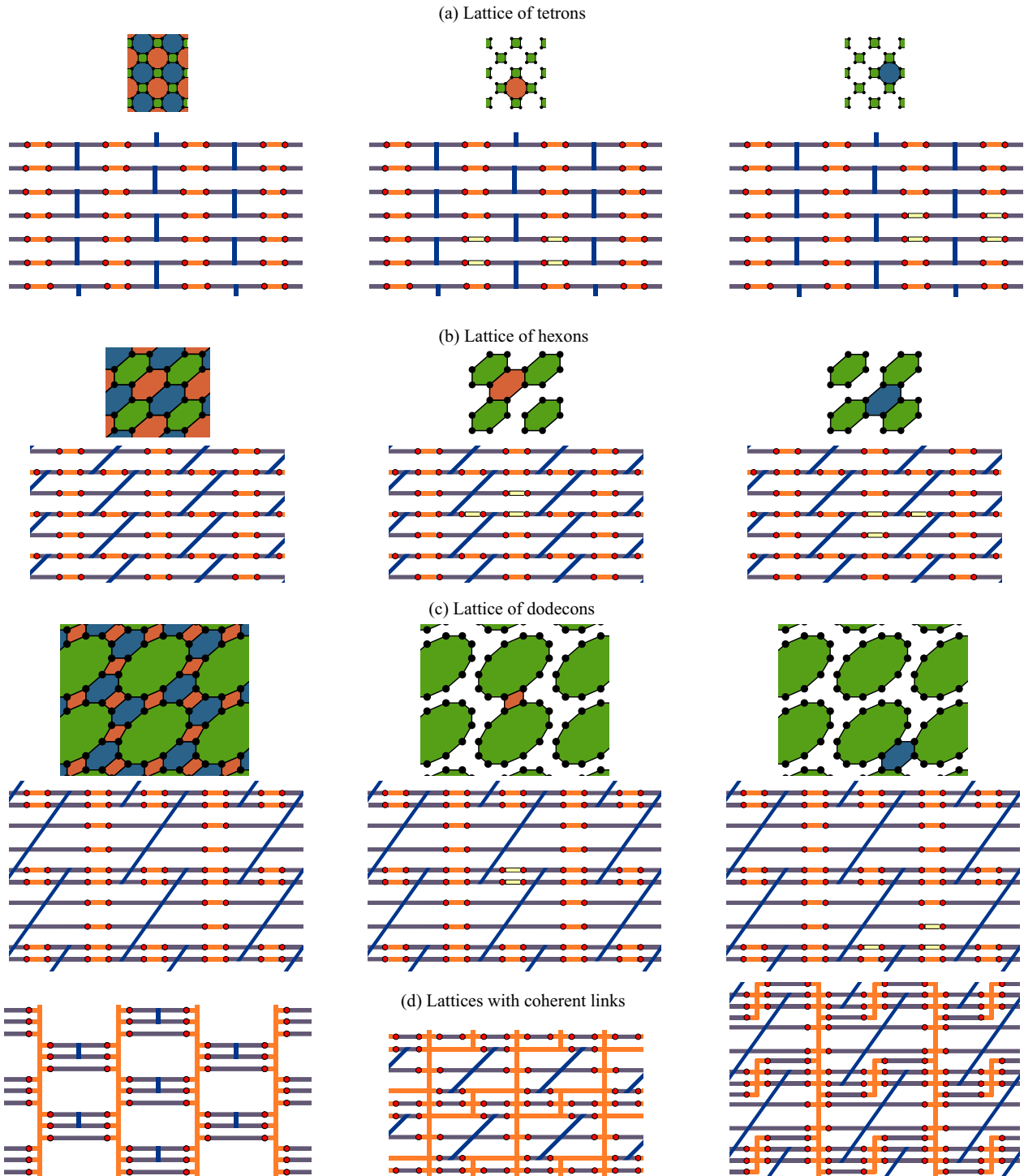


FIG. 27. Nanowire-based implementations of 4.8.8 (a), 6.6.6 (b), and 4.6.12 (c), with examples of tunnel-coupling configurations for red and blue stabilizer measurements. Coherent links (d) may be necessary for the measurement of other operators.

and tetron. Some of the physical hexons along the boundaries of the logical hexon and tetron have been replaced by physical octons. As a consequence, the dislocation line between the X boundary of the tetron and the Z boundary of the hexon no longer features any 8-Majorana operators, but only 6-Majorana operators. This reduces the maximum Majorana weight of

quantum computing with 6.6.6 Majorana surface codes to 6 Majoranas.

7. Twist-based lattice surgery with subsystem surface codes

In Fig. 25, we show the twist-based lattice surgery protocol [27] for a $d = 3$ subsystem surface code [30]. While all the

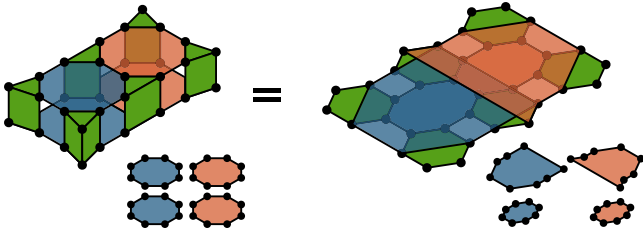


FIG. 28. 4.8.8 $([[6,2,2]]_m)$ Majorana color code in a 2D array of hexons.

check operators in the bulk of the code are weight-3, the measurement of the twist defect requires a five-qubit measurement. By replacing each qubit of the subsystem surface code with a tetron, one can see that the subsystem surface code resembles a 4.6.12 Majorana surface code, as is shown in Fig. 25.

APPENDIX B: QUANTUM-WIRE-BASED IMPLEMENTATIONS

1. Pauli product measurement protocol in a nanowire array

The Pauli product measurement protocol in Fig. 7 can be straightforwardly used to implement minimal-overhead Clifford gates in the wire-based architectures proposed in Ref. [33]. Essentially, hexons can be implemented as three topological superconducting nanowires in a parity sector that is fixed by a nontopological superconductor bridging the three wires. Moreover, in Fig. 26, tetrons correspond to two topological superconducting nanowires in a fixed-parity sector with an additional topological nanowire serving as a coherent link that is used for certain parity measurements. Products of Majorana operators are measured by opening tunnel couplings between topological superconductors and segments of a semiconducting nanowire network, such that the tunnel couplings form

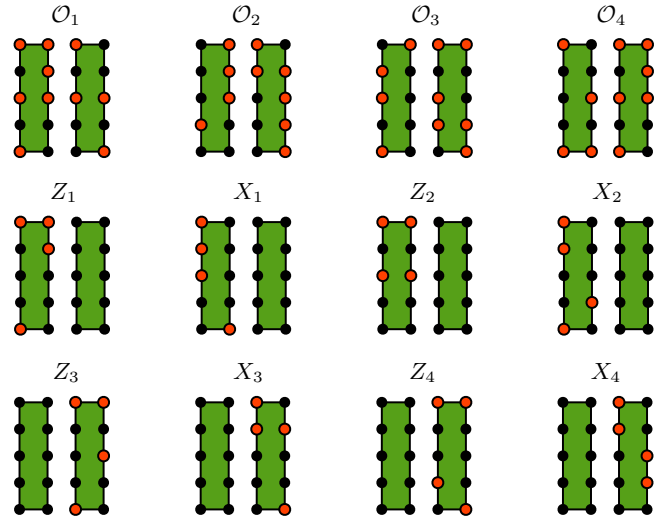


FIG. 30. Stabilizers \mathcal{O}_1 - \mathcal{O}_4 and logical operators of the $[[20,4,4]]_m$ code presented in Ref. [24].

closed paths. These semiconducting nanowire segments can be interpreted as quantum dots whose energy levels are shifted by the virtual process of an electron tunneling around the path, picking up all Majorana operators along the way. Therefore, suitable spectroscopy on the dots can be used to measure the product of these Majorana operators, as detailed in Ref. [33]. In Fig. 26, we show the tunnel-coupling configurations that implement the measurements that are part of the Pauli product measurement protocol. Note that some of the measurements require the use of coherent links, such that the Majorana weight of some measurements is increased. In particular, some of the 4-Majorana measurements in the second step become 6-Majorana measurements in this particular implementation.

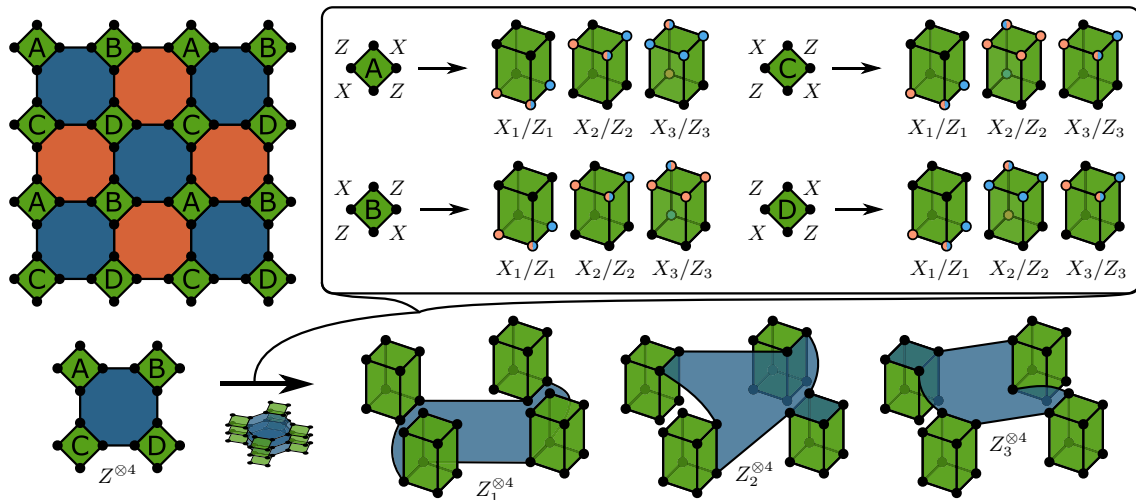


FIG. 29. Procedure to obtain a 4.8.8 $([[8,3,2]]_m)$ Majorana color code. In order to prevent stabilizer weights from increasing beyond 10 Majoranas, the tetrons of a 4.8.8 Majorana surface code are periodically labeled A, B, C, and D. Each tetron is replaced by a single octon, but the definition of the octon’s logical operators depends on the label of the corresponding tetron. In the figure, the octon’s Z (X) operators correspond to products of blue (red) Majoranas. The resulting stabilizers have a maximum weight of 10 Majoranas, as shown for the three overlapping blue Z type stabilizers.

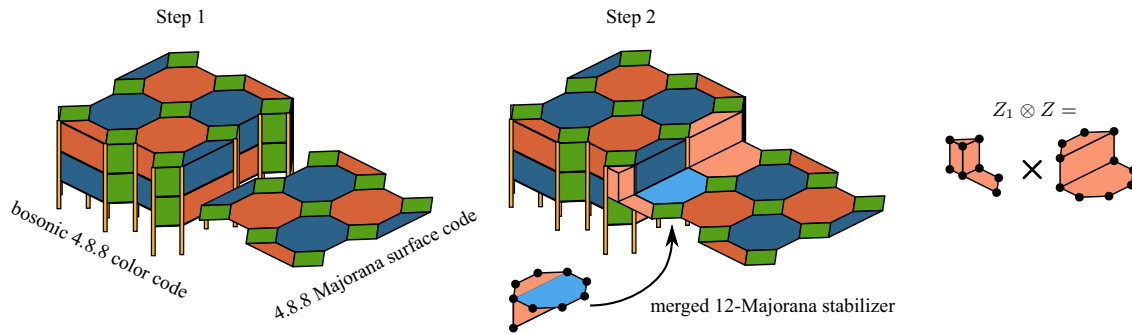


FIG. 31. Lattice surgery protocol for the measurement of the operator $Z_1 \otimes Z$ between a bosonic 4.8.8 color code tetron and a 4.8.8 Majorana surface code tetron. Z_1 is the Z operator of the first qubit encoded in the bosonic 4.8.8 color code tetron.

2. Majorana surface code implementations in a nanowire array

In Fig. 27, we show wire-based implementations of square lattices of tetrons (a), hexons (b), and dodecons (c), which can be used to implement 4.8.8, 6.6.6, and 4.6.12 Majorana surface codes, respectively. While we show the tunnel-coupling configurations to measure red and blue stabilizers, some irregular stabilizers, such as boundary stabilizers, dislocation lines, twist defect, or 2-Majorana operators for initialization and readout, may require the use of coherent links. Nanowire arrays that include coherent links are shown in Fig. 27(d). Note that operator measurements that require the use of coherent links have an increased Majorana weight compared to the optimal weight discussed in the main text. We also remark that, in principle, all stabilizers of one color can be measured simultaneously.

APPENDIX C: MAJORANA COLOR CODES

1. 4.8.8 ($[[6,2,2]]_m$) Majorana color code in a 2D array of hexons

Even though the Majorana color codes in Figs. 14 and 13 are drawn as three-dimensional codes, they can be implemented in 2D arrays of Majorana building blocks. In Fig. 28, this is shown for the example of a 4.8.8 ($[[6,2,2]]_m$) Majorana color code, which can be implemented in a 2D array of physical hexons. Note that, in contrast to Majorana surface codes, the red and blue check operators of Majorana color codes spatially overlap.

2. 4.8.8 ($[[8,3,2]]_m$) Majorana color code

A 4.8.8 ($[[8,3,2]]_m$) Majorana color code can be obtained by replacing each tetron of a 4.8.8 Majorana surface code with an $[[8,3,2]]_m$ code, i.e., an octon. Even though this code has a Majorana distance of $d_m = 2$, one of its logical operators is a 4-Majorana operator. In order to prevent the maximum stabilizer weight after concatenation from increasing to 16 Majoranas, the procedure shown in Fig. 29 can be used. Here, each surface code tetron is assigned a label A, B, C, or D. While each tetron is still replaced by an octon, the definition

of the octon's logical operators depends on the label. For A, B, C, or D type octons, the 4-Majorana logical operator is chosen to be Z_3 , X_3 , X_2 , or Z_2 , respectively. This guarantees that the stabilizers have a maximum weight of 10 Majoranas after concatenation. The figure shows an example of a $Z^{\otimes 4}$ stabilizer, which is replaced by three overlapping $Z_1^{\otimes 4}$, $Z_2^{\otimes 4}$, and $Z_3^{\otimes 4}$ stabilizers after concatenation. The operators $Z_2^{\otimes 4}$ and $Z_3^{\otimes 4}$ have a weight of 10 Majoranas.

3. Stabilizers and logical operators of the $[[20,4,4]]_m$ code

Figure 30 shows the stabilizers and logical operators of the $[[20,4,4]]_m$ code presented in Ref. [24]. The building blocks of the code are two decons, which are 10 Majoranas in a fixed-parity sector. All logical operators have a weight of 4 Majoranas. This means that a code obtained by concatenating a Majorana surface code with the $[[20,4,4]]_m$ code is an order-4-Majorana color code with a maximum stabilizer weight of 16 Majoranas.

4. Surface-to-color code lattice surgery

Figure 31 shows an example of a surface-to-color code lattice surgery in the spirit of Ref. [20]. The example shows the measurement of the product $Z_1 \otimes Z$ between the first qubit encoded in a bosonic 4.8.8 color code tetron and a 4.8.8 Majorana surface code. The blue 4-Majorana operator at the boundary of the surface code is merged with a red 8-Majorana operator at the boundary of the color code to yield the light blue 12-Majorana operator. New 8- and 12-Majorana operators (light red) are introduced whose product is precisely $Z_1 \otimes Z$. Such a scheme can always be used to measure the product between a Pauli operator encoded in a color code qubit and a Pauli operator of a surface code. By using color codes for data qubits encoded in logical hexons and surface codes for ancillary tetron qubits, the Pauli product measurement scheme of Sec. III can be implemented with a lower-space overhead compared to surface codes due to the more compact encoding of color codes, while at the same time benefiting from the low Majorana weight of surface-to-color code lattice surgery.

[1] A. Y. Kitaev, Unpaired Majorana fermions in quantum wires, *Sov. Phys. Usp.* **44**, 131 (2001).

[2] A. Y. Kitaev, Fault-tolerant quantum computation by anyons, *Ann. Phys. (NY)* **303**, 2 (2003).

- [3] J. Alicea, New directions in the pursuit of Majorana fermions in solid state systems, *Rep. Prog. Phys.* **75**, 076501 (2012).
- [4] C. W. J. Beenakker, Search for Majorana fermions in superconductors, *Annu. Rev. Condens. Matter Phys.* **4**, 113 (2013).
- [5] R. M. Lutchyn, E. P. A. M. Bakkers, L. P. Kouwenhoven, P. Krogstrup, C. M. Marcus, and Y. Oreg, Realizing Majorana zero modes in superconductor-semiconductor heterostructures, [arXiv:1707.04899](https://arxiv.org/abs/1707.04899).
- [6] R. Aguado, Majorana quasiparticles in condensed matter, *Riv. Nuovo Cimento* **11**, 523 (2017).
- [7] V. Mourik, K. Zuo, S. M. Frolov, S. R. Plissard, E. P. A. M. Bakkers, and L. P. Kouwenhoven, Signatures of Majorana fermions in in hybrid superconductor-semiconductor nanowire devices, *Science* **336**, 1003 (2012).
- [8] S. M. Albrecht, A. P. Higginbotham, M. Madsen, F. Kuemmeth, T. S. Jespersen, J. Nyg, P. Krogstrup, and C. M. Marcus, Exponential protection of zero modes in Majorana islands, *Nature (London)* **531**, 206 (2016).
- [9] M. Deng, S. Vaitiekėnas, E. Hansen, J. Danon, M. Leijnse, K. Flensberg, J. Nygård, P. Krogstrup, and C. Marcus, Majorana bound state in a coupled quantum-dot hybrid-nanowire system, *Science* **354**, 1557 (2016).
- [10] H. Zhang, C.-X. Liu, S. Gazibegovic, D. Xu, J. A. Logan, G. Wang, N. van Loo, J. D. S. Bommer, M. W. A. de Moor, and D. Car, Quantized Majorana conductance, *Nature (London)* **556**, 74 (2018).
- [11] D. Aasen, M. Hell, R. V. Mishmash, A. Higginbotham, J. Danon, M. Leijnse, T. S. Jespersen, J. A. Folk, C. M. Marcus, K. Flensberg, and J. Alicea, Milestones Toward Majorana-Based Quantum Computing, *Phys. Rev. X* **6**, 031016 (2016).
- [12] C. Knapp, T. Karzig, R. M. Lutchyn, and C. Nayak, Dephasing of Majorana-based qubits, *Phys. Rev. B* **97**, 125404 (2018).
- [13] B. M. Terhal, Quantum error correction for quantum memories, *Rev. Mod. Phys.* **87**, 307 (2015).
- [14] S. Bravyi, B. M. Terhal, and B. Leemhuis, Majorana fermion codes, *New J. Phys.* **12**, 083039 (2010).
- [15] S. Vijay and L. Fu, Physical implementation of a Majorana fermion surface code for fault-tolerant quantum computation, *Phys. Scr. T* **168**, 014002 (2016).
- [16] L. A. Landau, S. Plugge, E. Sela, A. Altland, S. M. Albrecht, and R. Egger, Towards Realistic Implementations of a Majorana Surface Code, *Phys. Rev. Lett.* **116**, 050501 (2016).
- [17] S. Plugge, L. A. Landau, E. Sela, A. Altland, K. Flensberg, and R. Egger, Roadmap to Majorana surface codes, *Phys. Rev. B* **94**, 174514 (2016).
- [18] Y. Li, Noise Threshold and Resource Cost of Fault-Tolerant Quantum Computing with Majorana Fermions in Hybrid Systems, *Phys. Rev. Lett.* **117**, 120403 (2016).
- [19] D. Litinski, M. S. Kesselring, J. Eisert, and F. von Oppen, Combining Topological Hardware and Topological Software: Color-Code Quantum Computing with Topological Superconductor Networks, *Phys. Rev. X* **7**, 031048 (2017).
- [20] D. Litinski and F. von Oppen, Braiding by Majorana tracking and long-range CNOT gates with color codes, *Phys. Rev. B* **96**, 205413 (2017).
- [21] S. Vijay, T. H. Hsieh, and L. Fu, Majorana Fermion Surface Code for Universal Quantum Computation, *Phys. Rev. X* **5**, 041038 (2015).
- [22] Y. Li, Fault-tolerant fermionic quantum computation based on color code, [arXiv:1709.06245](https://arxiv.org/abs/1709.06245).
- [23] S. Vijay and L. Fu, Quantum error correction for complex and Majorana fermion qubits, [arXiv:1703.00459](https://arxiv.org/abs/1703.00459).
- [24] M. B. Hastings, Small Majorana fermion codes, [arXiv:1703.00612](https://arxiv.org/abs/1703.00612).
- [25] H. Bombin, Topological Order with a Twist: Ising Anyons from an Abelian Model, *Phys. Rev. Lett.* **105**, 030403 (2010).
- [26] M. B. Hastings and A. Geller, Reduced space-time and time costs using dislocation codes and arbitrary ancillas, *Quantum Inf. Comput.* **15**, 962 (2015).
- [27] D. Litinski and F. von Oppen, Lattice surgery with a twist: Simplifying Clifford gates of surface codes, [arXiv:1709.02318](https://arxiv.org/abs/1709.02318).
- [28] H. Bombin and M. A. Martin-Delgado, Topological Quantum Distillation, *Phys. Rev. Lett.* **97**, 180501 (2006).
- [29] D. Gottesman, The Heisenberg representation of quantum computers, *Proc. XXII Int. Coll. Group. Th. Meth. Phys.* **1**, 32 (1999).
- [30] S. Bravyi, G. Duclos-Cianci, D. Poulin, and M. Suchara, Subsystem surface codes with three-qubit check operators, *Quantum Inf. Comput.* **13**, 963 (2013).
- [31] S. B. Bravyi and A. Y. Kitaev, Quantum codes on a lattice with boundary, [arXiv:quant-ph/9811052](https://arxiv.org/abs/quant-ph/9811052).
- [32] S. Plugge, A. Rasmussen, R. Egger, and K. Flensberg, Majorana box qubits, *New J. Phys.* **19**, 012001 (2017).
- [33] T. Karzig, C. Knapp, R. M. Lutchyn, P. Bonderson, M. B. Hastings, C. Nayak, J. Alicea, K. Flensberg, S. Plugge, Y. Oreg *et al.*, Scalable designs for quasiparticle-poisoning-protected topological quantum computation with Majorana zero modes, *Phys. Rev. B* **95**, 235305 (2017).
- [34] R. M. Lutchyn, J. D. Sau, and S. Das Sarma, Majorana Fermions and a Topological Phase Transition in Semiconductor-Superconductor Heterostructures, *Phys. Rev. Lett.* **105**, 077001 (2010).
- [35] Y. Oreg, G. Refael, and F. von Oppen, Helical Liquids and Majorana Bound States in Quantum Wires, *Phys. Rev. Lett.* **105**, 177002 (2010).
- [36] B. Criger and B. Terhal, Noise Thresholds for the $[[4, 2, 2]]$ -concatenated Toric Code, *Quantum Inf. Comput.* **16**, 1261 (2016).
- [37] H. Bombin, Topological subsystem codes, *Phys. Rev. A* **81**, 032301 (2010).
- [38] B. J. Brown, K. Laubscher, M. S. Kesselring, and J. R. Wootton, Poking Holes and Cutting Corners to Achieve Clifford Gates with the Surface Code, *Phys. Rev. X* **7**, 021029 (2017).
- [39] X.-G. Wen, Quantum Orders in An Exact Soluble Model, *Phys. Rev. Lett.* **90**, 016803 (2003).
- [40] P. O. Boykin, T. Mor, M. Pulver, V. Roychowdhury, and F. Vatan, New universal and fault-tolerant quantum basis, *Inf. Proc. Lett.* **75**, 101 (2000).
- [41] S. Bravyi and A. Kitaev, Universal quantum computation with ideal Clifford gates and noisy ancillas, *Phys. Rev. A* **71**, 022316 (2005).
- [42] C. Horsman, A. G. Fowler, S. Devitt, and R. V. Meter, Surface code quantum computing by lattice surgery, *New J. Phys.* **14**, 123011 (2012).
- [43] M. S. Kesselring, B. J. Brown, F. Pastawski, and J. Eisert, Boundaries and domain walls of the color code (unpublished).
- [44] P. Bonderson, D. J. Clarke, C. Nayak, and K. Shtengel, Implementing Arbitrary Phase Gates with Ising Anyons, *Phys. Rev. Lett.* **104**, 180505 (2010).

- [45] T. Karzig, Y. Oreg, G. Refael, and M. H. Freedman, Universal Geometric Path to a Robust Majorana Magic Gate, *Phys. Rev. X* **6**, 031019 (2016).
- [46] D. J. Clarke, J. D. Sau, and S. Das Sarma, A Practical Phase Gate for Producing Bell Violations in Majorana Wires, *Phys. Rev. X* **6**, 021005 (2016).
- [47] A. J. Landahl and C. Ryan-Anderson, Quantum computing by color-code lattice surgery, [arXiv:1407.5103](https://arxiv.org/abs/1407.5103).
- [48] Y. Li, A magic state's fidelity can be superior to the operations that created it, *New J. Phys.* **17**, 023037 (2015).
- [49] A. J. Landahl, J. T. Anderson, and P. R. Rice, Fault-tolerant quantum computing with color codes, [arXiv:1108.5738](https://arxiv.org/abs/1108.5738).
- [50] F. Pastawski and B. Yoshida, Fault-tolerant logical gates in quantum error-correcting codes, *Phys. Rev. A* **91**, 012305 (2015).
- [51] T. E. O'Brien, P. Rožek, and A. R. Akhmerov, Majorana-based fermionic quantum computation, [arXiv:1712.02353](https://arxiv.org/abs/1712.02353).
- [52] T. J. Yoder and I. H. Kim, The surface code with a twist, *Quantum* **1**, 2 (2017).
- [53] H. G. Katzgraber, H. Bombin, and M. A. Martin-Delgado, Error Threshold for Color Codes and Random Three-Body Ising Models, *Phys. Rev. Lett.* **103**, 090501 (2009).
- [54] R. S. Andrist, H. G. Katzgraber, H. Bombin, and M. A. Martin-Delgado, Error tolerance of topological codes with independent bit-flip and measurement errors, *Phys. Rev. A* **94**, 012318 (2016).
- [55] O. Zilberberg, B. Braunecker, and D. Loss, Controlled-NOT gate for multiparticle qubits and topological quantum computation based on parity measurements, *Phys. Rev. A* **77**, 012327 (2008).

4 | Color Codes in Tetron-Based Architectures

One of the techniques hinted at in the previous chapter was surface-to-color code lattice surgery. The following publication explains how bosonic color codes can be implemented in a tetron architecture, and how surface-to-color code lattice surgery can be used to decrease the overhead of long-range CNOT gates between color-code qubits.

Braiding by Majorana tracking and long-range CNOT gates with color codes

Daniel Litinski and Felix von Oppen

Dahlem Center for Complex Quantum Systems and Fachbereich Physik, Freie Universität Berlin, Arnimallee 14, 14195 Berlin, Germany

(Received 29 August 2017; published 8 November 2017)

Color-code quantum computation seamlessly combines Majorana-based hardware with topological error correction. Specifically, as Clifford gates are transversal in two-dimensional color codes, they enable the use of the Majoranas' non-Abelian statistics for gate operations at the code level. Here, we discuss the implementation of color codes in arrays of Majorana nanowires that avoid branched networks such as T junctions, thereby simplifying their realization. We show that, in such implementations, non-Abelian statistics can be exploited without ever performing physical braiding operations. Physical braiding operations are replaced by Majorana tracking, an entirely software-based protocol which appropriately updates the Majoranas involved in the color-code stabilizer measurements. This approach minimizes the required hardware operations for single-qubit Clifford gates. For Clifford completeness, we combine color codes with surface codes, and use color-to-surface-code lattice surgery for long-range multitarget CNOT gates which have a time overhead that grows only logarithmically with the physical distance separating control and target qubits. With the addition of magic state distillation, our architecture describes a fault-tolerant universal quantum computer in systems such as networks of tetrons, hexons, or Majorana box qubits, but can also be applied to nontopological qubit platforms.

DOI: [10.1103/PhysRevB.96.205413](https://doi.org/10.1103/PhysRevB.96.205413)

I. INTRODUCTION

The scalable fabrication of high-fidelity qubit platforms is necessary for large-scale quantum computation. In topological quantum computing [1–3], Majorana-based architectures [4–8] have been proposed as candidates for such high-fidelity qubits. Among the advantages that Majorana-based qubits may offer in comparison to conventional qubits are long coherence times, high-fidelity single-qubit Clifford gates by braiding, and ancilla-free stabilizer measurements for quantum error correction. Recent experiments have demonstrated considerable progress towards realizing Majorana zero modes (Majoranas) in topological superconductors [9–13], but topological qubits are yet to be implemented and their advantages remain to be confirmed experimentally.

As Majorana-based qubits are still expected to have a finite lifetime [14–17], quantum error correction is necessary for fault-tolerant quantum computation [18–25]. In a recent work [26], we have argued that Majorana-based qubits are particularly well suited for quantum error correction with topological color codes [27,28]. Unlike in surface codes [3,29,30], the Clifford gates are transversal in two-dimensional color codes. Thus logical Clifford gates are implemented on the code level by performing independent Clifford gates on all (pairs of) physical qubits that make up the logical qubit(s). Importantly, this transversal gate set enables the use of braiding for logical gates, thereby fully exploiting the topological protection provided by Majorana-based hardware. Moreover, the existence of transversal gates has additional advantages. Independent operations on the physical qubit do not spread errors during gate operations and minimize the time overhead by allowing parallel implementation.

We discussed a physical implementation of a Majorana color code which relies on topological superconductor networks, where Majoranas are braided by moving them through branched geometries. Moving Majoranas was also necessary for lattice surgery [31,32] and magic state distillation [33], which are required to complete the universal gate set. However,

current experiments where proximity-coupled quantum wires are driven into the topological phase require an external magnetic field in the direction of the wire. This might constitute a significant obstacle for all braiding protocols that rely on the movement of Majoranas in branched geometries [34–37]. Moreover, movement of Majoranas has also been shown to be susceptible to thermal noise [38]. To overcome these problems, recent works have proposed architectures that avoid T junctions, and are instead based on arrays of parallel nanowires [22,23,39–41]. In this work, we show that Majorana color codes can be naturally implemented in such setups, thereby entirely avoiding T junctions and explicit movement of Majoranas.

An important aspect of the recent works on implementing topological qubits in arrays of parallel wires is that braiding is no longer performed by moving Majoranas or coupling Majoranas in judicious ways, but instead by measuring a sequence of two-Majorana parity operators [41,42]. Thus these architectures shift the experimental challenge away from the fabrication of branched geometries and towards the access to measurements of various local Majorana parity operators. It was pointed out that, in this setting, single-qubit Clifford gates can be implemented by an entirely classical software-based procedure [41,43,44], which we refer to as *Majorana tracking*. This procedure obviates explicit hardware operations and, similar to Pauli tracking, only requires appropriate updates of the qubit's reference frame. We show that the transversal gates of color codes take Majorana tracking to the level of logical qubits, and thereby reduce the overhead of logical single-qubit Clifford gates to a (classical) minimum. The only required hardware operation is the measurement of certain local Majorana parity operators corresponding to the stabilizers of the quantum error-correcting code.

Universal fault-tolerant quantum computation can be achieved by implementing two more logical gates: the controlled-NOT (CNOT) gate and the T gate. While logical CNOTs can in principle be implemented transversally using physical CNOTs, this requires nonlocal physical gates. Instead,

it is more convenient to implement logical CNOTs via lattice surgery [31], which only requires local operations. Here, we present a scheme for logical CNOTs which combines color codes with surface code ancillas and employs color-to-surface-code lattice surgery [45]. This protocol implements the CNOT gate without the need for any movement of Majoranas while retaining the long-range communication between color-code qubits of our earlier implementation. This also provides us with a long-range multitarget CNOT, which is an essential part of magic state distillation protocols, thereby completing the universal gate set.

Here, we are mainly interested in providing a proof-of-principle implementation of Majorana-based color-code quantum computation in a network of tetrons [41]. However, it should be emphasized that the combination of Majorana-based hardware with color-code error correction transcends our specific implementations. It seems likely that this combination can be used to one's advantage in many, if not all, future implementations of fault-tolerant Majorana-based quantum computation. In fact, our lattice-surgery-based scheme can in principle be applied even to nontopological qubit architectures. Nevertheless, the robustness of physical single-qubit Clifford gates and the ease of stabilizer measurements are key advantages of Majorana-based qubit platforms.

II. MAJORANA TRACKING AND COLOR CODES

A Majorana-based qubit can be defined using three Majorana fermions γ_1 , γ_2 , and γ_3 , with $\{\gamma_i, \gamma_j\} = 2\delta_{i,j}$ and $\gamma_i = \gamma_i^\dagger$. Since Majorana fermions in physical systems always come in pairs, it is convenient to define the qubit using four Majoranas with fixed total parity $-\gamma_1\gamma_2\gamma_3\gamma_4 = 1$, such that all two-Majorana parity operators $i\gamma_m\gamma_n$ of a qubit can be expressed in terms of the first three Majoranas. In the Schrödinger picture, a qubit is defined using two computational states $|0\rangle$ and $|1\rangle$ in the σ_z basis. For our purposes, it will be instead more useful to express the qubit in the Heisenberg picture, where we define the qubit by its σ_x - and σ_z -Pauli operators, which in their default state are

$$\sigma_z = i\gamma_1\gamma_2, \quad \sigma_x = i\gamma_2\gamma_3. \quad (1)$$

Consequently, the remaining Pauli operator is $\sigma_y = i\gamma_1\gamma_3$. One can check that these operators square to unity and fulfill the commutation relations of Pauli operators $[\sigma_i, \sigma_j] = 2i\epsilon_{ijk}\sigma_k$. Two Pauli operators are sufficient to define a qubit, as any single-qubit unitary operator can be expressed in terms of σ_x and σ_z via the Euler decomposition.

The basic framework of our architecture is nanowire arrays, which are two-dimensional networks of Majorana-based physical qubits. Several proposals for implementations of such nanowire arrays can be found in Refs. [22,23,39–41]. Based on these proposals, we assume that the following basic operations can be implemented in the nanowire array.

(i) Measurements of local $2n$ -Majorana fermion parity operators $i^n \prod_{i=1}^{2n} \gamma_i$.

(ii) Some nonrobust implementation of a possibly faulty T gate ($\pi/8$ gate) on physical qubits.

As already emphasized above, we do *not* require that the Majoranas can be moved through the network.

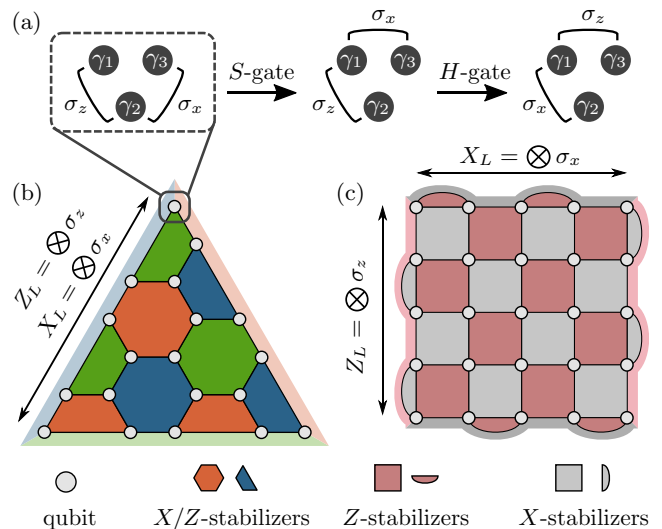


FIG. 1. (a) Majorana-based qubit consisting of three Majoranas and an example for Majorana tracking. Starting from the default encoding $\sigma_z = i\gamma_1\gamma_2$ and $\sigma_x = i\gamma_2\gamma_3$, an S gate changes the encoding to $\sigma_z = i\gamma_1\gamma_2$ and $\sigma_x = i\gamma_1\gamma_3$. A subsequent H gate changes it to $\sigma_z = i\gamma_1\gamma_3$ and $\sigma_x = i\gamma_1\gamma_2$. Keeping track of the current encoding for each physical qubit via a classical computer is referred to as Majorana tracking. (b) Triangular color-code qubits can be defined on a hexagonal lattice, where each vertex is a Majorana-based qubit comprised of three Majoranas (or four Majoranas with fixed total parity). Each face corresponds to two stabilizers $\sigma_z^{\otimes m}$ and $\sigma_x^{\otimes m}$. Products of σ_z and σ_x operators along any one of the three boundaries correspond to logical Z_L and X_L operators. (c) In surface code qubits, on the other hand, the support of X and Z stabilizers does not coincide, and the two different edges correspond to the Z_L and X_L operators, respectively.

A. Physical single-qubit Clifford gates: Majorana tracking

The first operation includes the measurement of all two-Majorana parity operators—and therefore all Pauli operators—of a physical qubit. This enables the use of Majorana tracking for a particularly simple implementation of the single-qubit Clifford gates as pioneered in Refs. [41,43]. These gates map Pauli operators onto other Pauli operators and are products of Hadamard (H) and phase (S) gates. Specifically, the action of these two gates on the Pauli operators is

$$\begin{aligned} H : \quad \sigma_z &\rightarrow \sigma_x, \quad \sigma_x \rightarrow \sigma_z, \\ S : \quad \sigma_z &\rightarrow \sigma_z, \quad \sigma_x \rightarrow i\sigma_x\sigma_z. \end{aligned} \quad (2)$$

Since the H and S gates can be implemented by braiding, their application simply redefines the Majoranas involved in the corresponding two-Majorana parity operator. Thus, instead of physically braiding Majoranas, one can alternatively keep track of the Majorana operators that define the σ_z and σ_x of each physical qubit using a classical computer. In analogy to Pauli tracking [46], we refer to this procedure as Majorana tracking.

As a concrete example, consider the sequence of operations shown in Fig. 1(a). Starting from the default encoding in Eq. (1), an S gate takes the encoding to $\sigma_z = i\gamma_1\gamma_2$ and $\sigma_x = i\gamma_1\gamma_3$. A subsequent H gate will exchange these two operators

to $\sigma_z = i\gamma_1\gamma_3$ and $\sigma_x = i\gamma_1\gamma_2$. So instead of initializing the qubit in a σ_z eigenstate, physically performing the two gates, and then reading out the qubit in the σ_z basis, one can simply initialize the qubit in a σ_z eigenstate and then measure the $\sigma_y = i\gamma_1\gamma_3$ operator.

It should not be surprising that Clifford gates can be treated entirely classically, as these gates can be efficiently simulated on a classical computer by virtue of the Gottesman-Knill theorem [47]. As this classical tracking of Pauli operators can also be done with nontopological, e.g., superconducting, qubits, it would appear that Majorana tracking does not utilize braiding. However, conventional qubits still require a hardware operation for the rotation of the Pauli basis during readout. While for conventional qubits the angle of rotation is susceptible to errors, with Majorana tracking the angle is robust. Even though Majorana tracking eliminates any hardware operation for single-qubit Clifford gates, it leads to the same robust gates as braiding. In this sense, Majorana tracking *is* braiding.

Therefore, Majorana tracking can also be used to probe the non-Abelian statistics of Majorana zero modes. With Majorana tracking, a fusion-rule detection experiment in the spirit of Ref. [37] would correspond to alternating measurements of σ_z and σ_x . If Majorana zero modes are present, the measurement results will be entirely uncorrelated, whereas repeated measurements of σ_z will always yield the same result. In this way, the fusion-rule detection experiment probes the robustness of the single-qubit Clifford gates.

B. Logical single-qubit Clifford gates: Color codes

The gates that can be implemented by Majorana tracking are physical gates on physical qubits. However, these Majorana-based qubits only have a finite lifetime which is set by processes that introduce errors, such as quasiparticle poisoning. In order to quantum compute beyond the coherence time of physical qubits, a quantum error-correcting code needs to be used. This allows for *fault-tolerant* quantum computing, which relies on combining many physical qubits into one logical qubit. This not only replaces the physical error rate by an (in principle) arbitrarily low logical error rate, but also substitutes physical gates with logical gates.

It is desirable to use Majorana tracking for logical gates in order to minimize the overhead of single-qubit Clifford gates. But this can only be done if these gates are transversal gates of the error-correcting code, i.e., if the logical H and S gates are $H_L = H^{\otimes n}$ and $S_L = S^{\otimes n}$, where n is the number of physical qubits in a logical qubit. This is precisely the reason why color codes are a natural choice for Majorana-based qubits [26], as their set of transversal gates are the Clifford gates.

Using triangular color codes [27,28], a logical qubit is encoded by n physical qubits located at the vertices of the triangle with hexagonal tiling shown in Fig. 1(b). The figure shows a specific qubit with code distance $d = 5$, but this construction can be generalized to arbitrary odd code distances. As described above, each physical qubit effectively corresponds to three Majorana fermions. The logical qubit is initialized in the logical $|0_L\rangle$ state by initializing the n physical qubits in the $|0\rangle$ state by measuring $i\gamma_1\gamma_2$ of all physical qubits, and then measuring the stabilizers of the code. These

$n - 1$ stabilizers are defined by the faces of the hexagonally tiled triangle, with each face defining an X -type stabilizer $O_X = \sigma_x^{\otimes m}$ and a Z -type stabilizer $O_Z = \sigma_z^{\otimes m}$, where m is the number of qubits that are part of a face. In analogy to Majorana surface codes [22,23], one can represent color codes as Majorana fermion codes by identifying $\sigma_z = i\gamma_1\gamma_2$ and $\sigma_x = i\gamma_2\gamma_3$. Thus the stabilizers in Fig. 1(b) are products of 8 or 12 Majorana fermions. A color-code qubit can be read out in any Pauli basis by measuring all physical qubits in the corresponding basis.

Quantum error-correcting codes typically operate in cycles. In each code cycle, the stabilizers are measured to determine the error syndrome, errors are corrected, and logical gate operations are performed. The single-qubit logical Clifford gates are transversal in color codes, i.e., a logical H gate corresponds to physical H gates on all qubits, whereas a logical S gate is a combination of physical S and S^\dagger gates. For instance, a conventional procedure for a logical S gate would be to measure and correct the error syndrome, transversally perform physical S and S^\dagger gates (e.g., by braiding), and again measure the error syndrome and correct errors. With Majorana tracking, the physical gate operations are replaced by an update of the σ_z and σ_x operators of all physical qubits. While the σ_z operators are unaffected by the S and S^\dagger gates, the σ_x operators are changed from $i\gamma_2\gamma_3$ in the default encoding to $\pm i\gamma_1\gamma_3$. In other words, Majorana tracking modifies which Majorana fermions are part of the stabilizer measurements. In the case of an S gate, the X -type stabilizers $\sigma_x^{\otimes 6}$ are replaced by Y -type stabilizers $\sigma_y^{\otimes 6}$ in the following rounds of syndrome measurement, i.e., the X -type stabilizers are changed from products of $i\gamma_2\gamma_3$ to products of $i\gamma_1\gamma_3$.

In this way, keeping track of the current Majorana composition of σ_z and σ_x for each physical qubit implements logical single-qubit Clifford gates with Majorana color codes. However, considering that, in the default encoding, the measurement of X - and Z -type stabilizers automatically measures the Y -type stabilizers as their product, it is not necessary to actually change the measured stabilizers after the application of single-qubit Clifford gates. This is due to the fact that the support of X and Z stabilizers of color-code qubits coincides. Instead, Majorana tracking on the level of logical qubits, similar to the tracking procedure on physical qubits, merely updates the Majoranas measured during qubit readout. In the previous example of a logical S gate, tracking changes the X_L -basis readout from a measurement of $i\gamma_2\gamma_3$ to the measurement of $i\gamma_1\gamma_3$ of *all* physical qubits. An appropriate update of certain stabilizer operators will be necessary as soon as CNOT gates are involved, because this introduces X and Z stabilizers whose support does *not* coincide, as we discuss in Sec. IV A.

Previously [26], we argued that Majorana-based qubits and color codes are a natural fit, as transversal Clifford gates allow for the use of braiding for logical gates. In the context of the present work, this statement takes the equivalent form: owing to braiding by Majorana tracking, Majorana-based qubits can be read out in every Pauli basis without the need for intermediate hardware operations. Similarly, due to transversal Clifford gates, color-code qubits can be measured in every logical Pauli basis without requiring intermediate logical gates. Thus color codes in combination with Majorana-based qubits

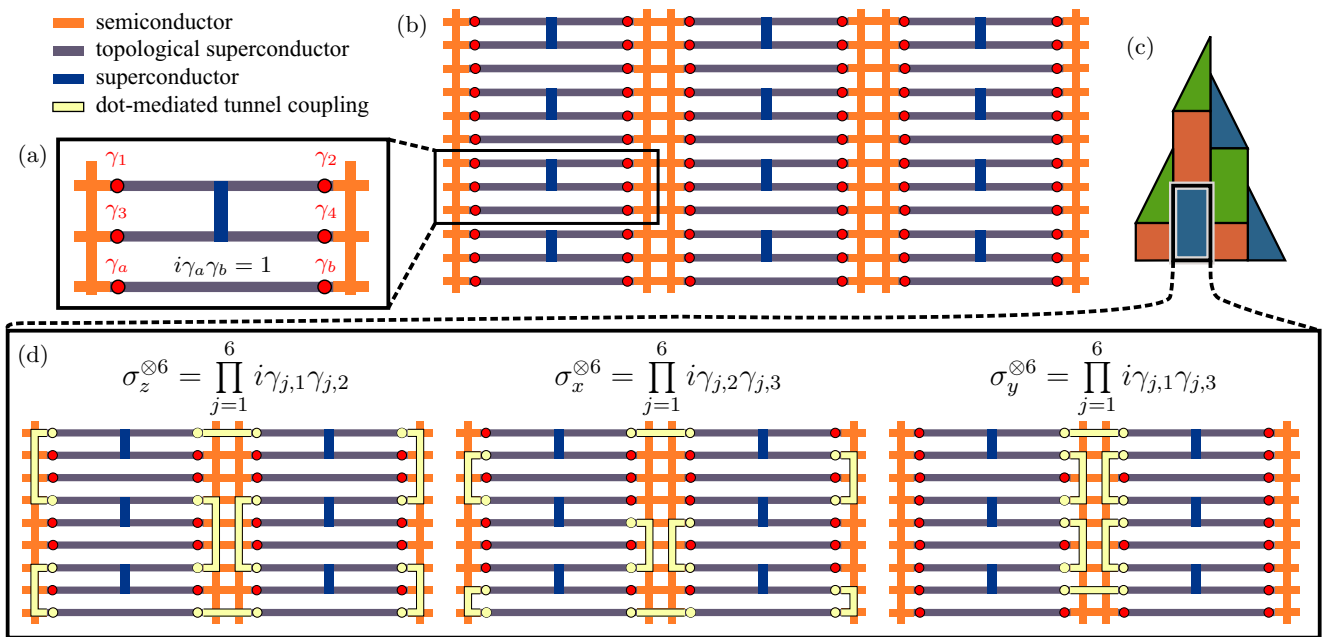


FIG. 2. (a) Single tetron [41] consists of two topological superconducting nanowires hosting four Majoranas $\gamma_1 \dots \gamma_4$. The two wires are bridged by an ordinary superconductor which fixes the total parity sector $-\gamma_1\gamma_2\gamma_3\gamma_4$. In addition, a coherent link formed by a topological superconducting nanowire hosting Majoranas γ_a and γ_b with a fixed parity is part of the basic building block. The three Majorana nanowires are connected to a semiconducting nanowire network via gate-tunable tunnel couplings. (b) A network of tetrons forms a square lattice of physical qubits. (c) In such a square lattice, a triangular color-code qubit can be defined in a brick wall geometry. (d) Configurations of the tunnel couplings used to measure three different stabilizers, which are either products of $\gamma_1\gamma_2$ of each tetron, or $\gamma_2\gamma_3$, or $\gamma_1\gamma_3$. One can verify that, in all three cases, the circular paths only contain the corresponding Majoranas of each tetron and Majoranas that belong to coherent links.

reduce the overhead of *logical* single-qubit Clifford gates to a minimum.

III. IMPLEMENTATION WITH TETRONS

In this section, we present a proof-of-principle implementation of a Majorana color code in a nanowire array, which differs from our earlier setup [26] in two essential ways. First, the present implementation relies on recent suggestions to realize Majorana-based topological qubits using only parallel topological superconducting nanowires. Second, as a consequence of implementing braiding at the code level by Majorana tracking, Majoranas no longer need to be moved within the network.

Specifically, we present an implementation in a network of tetrons. A tetron [39,41] is a qubit [Fig. 2(a)] that consists of two topological superconducting nanowires with four Majoranas $\gamma_1 \dots \gamma_4$ with fixed total parity $-\gamma_1\gamma_2\gamma_3\gamma_4$. The fixed parity sector not only protects the qubit from quasiparticle poisoning, but also enables the use of the fourth Majorana γ_4 for quantum computation. In the even parity sector, we can identify $\sigma_z = i\gamma_1\gamma_2 = i\gamma_3\gamma_4$ and $\sigma_x = i\gamma_2\gamma_3 = i\gamma_1\gamma_4$. Furthermore, each tetron contains a third floating Majorana nanowire with Majoranas γ_a and γ_b and fixed parity $i\gamma_a\gamma_b$ acting as a coherent link. Gate-tunable tunnel couplings connect the three topological superconducting nanowires to a semiconductor network. The network of tetrons shown in Fig. 2(b) corresponds to the architecture described in Ref. [41], but with two vertical semiconductor wires between adjacent tetrons, instead of just one. (However, this is not a requirement,

as the implementation of a color code is also possible in the setup described in Ref. [41]; see Appendix A.) The tetron qubits form a square lattice which can be used to encode color-code qubits in a brick wall geometry; see Fig. 2(c).

With tetrons, $2n$ -Majorana parity operators are measured by opening tunnel couplings between tetrons such that they form a closed path, as discussed in Ref. [41]. The semiconducting segments that couple neighboring tetrons form quantum dots. Their energy levels are shifted by virtual processes that tunnel electrons around this closed path. As these processes involve each Majorana operator along this path exactly once, the energy shift depends on the product of the Majoranas, i.e., on the $2n$ -Majorana parity. Suitable spectroscopy on the dots can thus be used to measure this parity. Essentially, this measures the product of all Majoranas along a closed loop formed by the gate-tunable tunnel couplings, thereby implementing local $2n$ -Majorana parity measurements.

Consider the configuration of the tunnel couplings in the left panel of Fig. 2(d). The circular path formed by the coupled tetrons involves the γ_1 and γ_2 Majoranas of each tetron, and four Majoranas that belong to coherent links. Since the parity of the coherent links is known, this configuration can be used to measure the 12-Majorana operator that corresponds to $\sigma_z^{\otimes 6}$ in the default encoding. The center panel shows a configuration that measures the product of γ_2 and γ_3 Majoranas of each tetron, corresponding to a $\sigma_x^{\otimes 6}$ operator in the default encoding. Note that since the total parity sector of each tetron is fixed, $i\gamma_2\gamma_3 = i\gamma_1\gamma_4$. It is also possible to measure $\sigma_y^{\otimes 6}$ stabilizers, whose configuration is shown in the right panel of Fig. 2(d) and does not require the use of coherent links.

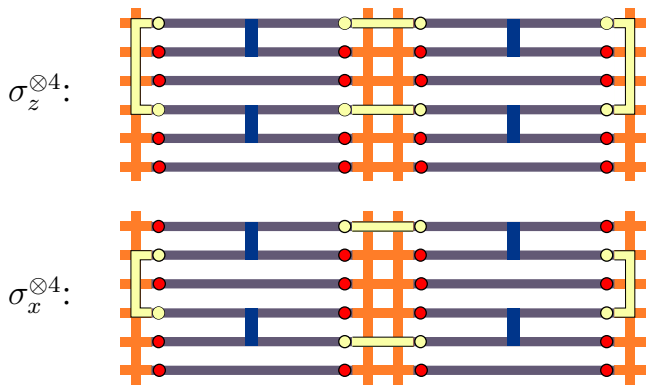


FIG. 3. Tetron tunnel coupling configurations for the measurement of the four-qubit stabilizers of the surface code.

IV. UNIVERSAL QUANTUM COMPUTATION

Having discussed logical single-qubit Clifford gates, two more gates are required for universal quantum computation: a logical controlled-not (CNOT) gate and a logical *T* gate, where $T = \exp(i\sigma_z\pi/8)$. As discussed in Ref. [26] for topological superconductor networks with branched geometries, these operations can be implemented by lattice surgery and magic state distillation, respectively. Here, we adapt this scheme to architectures where Majoranas cannot be moved, such that only the aforementioned stabilizer measurements and physical *T* gates are required.

A. Long-range CNOT gates

A logical CNOT gate between two color-code qubits can be implemented using lattice surgery with the help of an ancilla qubit [32]. This scheme effectively realizes the circuit identity shown in Fig. 4(a) using an ancilla qubit initialized in the σ_x eigenstate $|+\rangle$. A CNOT gate corresponds to a ZZ-parity ($\sigma_z \otimes \sigma_z$) measurement between the control qubit and the ancilla, a subsequent XX-parity measurement between ancilla and target, and a final σ_z measurement of the ancilla qubit. Note that the protocol requires parity measurements between *logical* qubits. Lattice surgery is a fault-tolerant protocol for such parity measurements requiring only the measurement of additional stabilizer operators straddling the

adjacent boundaries of two logical qubits. Essentially, lattice surgery measures the product of the logical operators defined on the two boundaries.

These boundary operators depend on the kind of logical qubit that is used. Triangular color-code qubits have three boundaries: a red, a green, and a blue edge. Strings of σ_z and σ_x operators along any of these edges are logical Z_L and X_L operators, respectively, as illustrated in Fig. 1(b). Surface code qubits, on the other hand, have pairs of opposing *X* and *Z* edges, also referred to as rough and smooth edges, and are drawn as gray and purple edges in Fig. 1(c). The logical Z_L operator is a product of σ_z operators along any of the two purple boundaries, whereas X_L is a string of σ_x operators along a gray boundary. With tetrons, the measurement of the four-qubit surface code stabilizer operators $\sigma_z^{\otimes 4}$ and $\sigma_x^{\otimes 4}$ is similar to the color-code stabilizer measurements, as shown in Fig. 3. In the following protocols, we use surface codes instead of color codes to encode ancilla qubits, as the CNOT protocol does not require the use of any transversal Clifford gates on the ancillas. Apart from lattice surgery, the only required surface code operations are the initialization in a σ_x -basis eigenstate, and a σ_z -basis measurement, both of which amount to σ_x and σ_z measurements of all physical qubits, and to stabilizer measurements. The main advantage of using surface code ancillas for CNOTs between color-code qubits is that, compared to color-code ancillas, they require fewer qubits and feature lower-weight stabilizers, as we discuss in Appendix B.

We first discuss logical CNOT gates between neighboring color-code qubits; see Fig. 4(b). The shape of the surface code qubit is chosen such that one *Z* boundary is adjacent to the control qubit and an *X* boundary is next to the target qubit. In the first step (b2), the *X* stabilizers along the boundary with the control qubit are merged to form six-qubit stabilizers (dark gray) and new *Z* stabilizers (light purple) are introduced. While this is not evident from the figure, the boundary *Z* stabilizers of the color-code qubit remain unchanged. In the new configuration (b2), all stabilizers commute, and the number of stabilizers has increased by one, i.e., one bit of information is measured. As the gray boundary stabilizers are merely the product of the previously known boundary stabilizers, the only nontrivial measurement outcome is given by the purple boundary stabilizers. Since they contain each boundary qubit exactly once, their product is precisely the

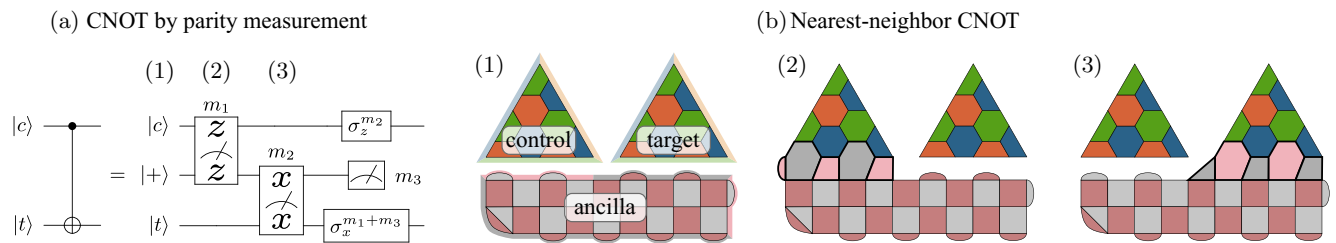


FIG. 4. (a) Quantum circuit corresponding to the logical CNOT gate between a control $|c\rangle$ and target $|t\rangle$ by lattice surgery, using an ancilla qubit initialized in the $|+\rangle$ state. First, the ZZ parity $\sigma_z \otimes \sigma_z$ between control and ancilla is measured. Next, the XX parity $\sigma_x \otimes \sigma_x$ between ancilla and target is measured, and the ancilla is read out in the σ_z basis. The three measurement outcomes are used to determine a final Pauli correction. (b) Nearest-neighbor CNOT between color-code qubits using a surface code ancilla. Lattice surgery between the green color-code boundary and the purple surface code boundary (b2) measures the ZZ parity, whereas surgery with the gray surface code boundary (b3) constitutes an XX-parity measurement.

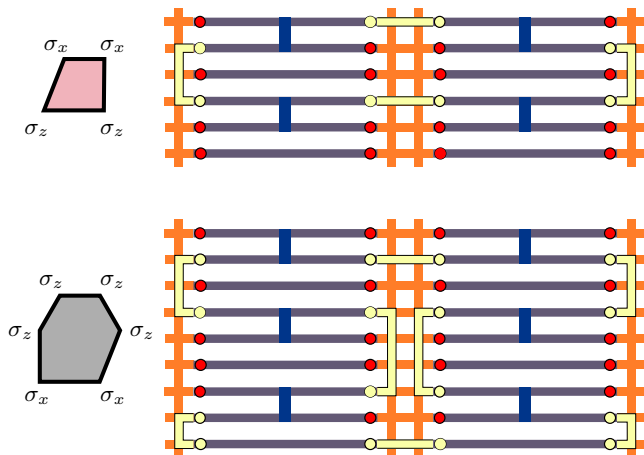


FIG. 5. Tetron tunnel coupling configurations to measure the lattice surgery boundary stabilizers between control and ancilla qubit for the case of a preceding H gate on the control qubit. The measured operators are $\sigma_x^{\otimes 2} \otimes \sigma_z^{\otimes 2}$ (light purple) and $\sigma_z^{\otimes 4} \otimes \sigma_x^{\otimes 2}$ (dark gray). This effectively describes a logical $\sigma_x \otimes \sigma_z$ measurement between control and ancilla. This protocol can be straightforwardly adapted to measure any other product of two logical Pauli operators.

ZZ parity between control and ancilla. Thus lattice surgery provides a fault-tolerant logical parity measurement. Similarly, in the next step (b3), lattice surgery merges the Z stabilizers along the X boundary of the surface code ancilla and a boundary of the target qubit. The product of gray boundary X stabilizers yields the XX parity. Finally, the surface code ancilla can be measured in the σ_z basis by measuring all physical qubits and applying classical error correction, thereby completing the protocol of Fig. 4(a).

The role of Majorana tracking in this protocol is to appropriately update the composition of the stabilizers measured during lattice surgery. The protocol introduces X and Z stabilizers with noncoinciding support along the boundaries of the qubits. Thus these stabilizers need to be appropriately updated by the tracking procedure described in Sec. II B. For instance, a preceding H gate on the control qubit in Fig. 4 would change the light purple boundary stabilizers in (b1) from $\sigma_z^{\otimes 4}$ to $\sigma_x^{\otimes 2} \otimes \sigma_z^{\otimes 2}$. Accordingly, tracking would also change the dark gray boundary stabilizers to $\sigma_z^{\otimes 4} \otimes \sigma_x^{\otimes 2}$ and the red four-qubit boundary stabilizers of the control qubit to $\sigma_x^{\otimes 4}$. In Fig. 5, we show the tetron tunnel coupling configurations used to measure these updated stabilizers for this particular example, but the procedure straightforwardly generalizes to all other possible cases. An update for the nonboundary color-code stabilizers unaffected by lattice surgery can still be avoided, as their X - and Z -stabilizer support still coincides during the lattice surgery protocol.

Surface code ancillas are also useful for CNOT gates between color-code qubits that are far away from each other. Lattice surgery can be used to measure the ZZ parities between the control qubit and multiple ancilla qubits simultaneously [31], thereby initializing multiple ancillas at the same time. Consider the situation in Fig. 6(a), where the distance between two separated color-code qubits is bridged by two surface code ancillas. Lattice surgery (a2) can simultaneously measure the

ZZ parities between control and first ancilla and between both ancillas. This is equivalent to parity measurements between control and both ancillas, as the ZZ parity between control and second ancilla is given by the product of both measurements. Since the two Z boundaries of the long ancilla are at opposite ends of the qubit, this lattice surgery step prepares an ancilla qubit adjacent to the distant target qubit for the next XX -parity measurement. Thus this protocol yields a long-range CNOT gate between arbitrarily distant qubits with essentially the same time overhead as the nearest-neighbor CNOT. The unused long ancilla qubit cannot be discarded right away, as it is still entangled with the control qubit, but needs to be read out in the σ_x basis with outcome m by measuring all physical qubits in the σ_x basis, leading to a σ_z^m correction on the control qubit.

B. Multitarget CNOTs for magic state distillation

Clifford gates and physical T gates are sufficient for universal quantum computing. One type of protocol using these ingredients for *logical* T gates is magic state distillation, whose precision scales with the protocol length. In such protocols, a physical magic state is initialized by applying a physical T gate to a physical qubit in the $|+\rangle$ state. With tetrons [41], physical T gates can be implemented via a measurement-based analog of the parity echo protocol introduced in Ref. [48]. The resulting physical magic state is converted into a (faulty) logical magic state by code injection [26,32], which requires only stabilizer measurements. Magic state distillation protocols convert many faulty magic states into fewer magic states with higher fidelity. Typically, these protocols rely on multitarget CNOT gates, i.e., CNOTs with one control qubit but multiple target qubits. For instance, the circuit corresponding to the 15-to-1 distillation protocol [28,33] consists of 34 CNOT gates. But since many of these CNOTs have the same control qubit, the protocol actually requires only five *multitarget* CNOTs.

Fortunately, lattice surgery can be used to implement multitarget CNOTs with the same time overhead as single CNOTs, as we show in Fig. 6(b). Here, lattice surgery measures the ZZ parities between the control and each ancilla qubit (b2). Therefore, each of the ancillas is treated like an ancilla qubit after step (2) of the protocol in Fig. 4(a), but for multiple simultaneous CNOT protocols. After the XX -parity measurements between ancillas and their targets (b3), the ancillas that were used for CNOTs are read out in the σ_z basis, whereas the ancillas that were used to bridge long distances are read out in the σ_x basis.

In this way, we establish color-to-surface code lattice surgery as a useful tool for fault-tolerant long-range multitarget CNOT gates between color-code qubits. Importantly, for a fixed code distance, the overhead of our protocol scales very favorably with the control-target separation s . As to the space overhead, strings of errors that connect the gray edges (or X boundaries) of the long surface code qubit during the measurement of the ZZ parities can lead to errors in the CNOT protocol. While these error strings are suppressed exponentially in the width of the surface code qubits, the number of possible strings grows linearly with their length. Thus the width needs to increase with $O(\ln s)$ in order to

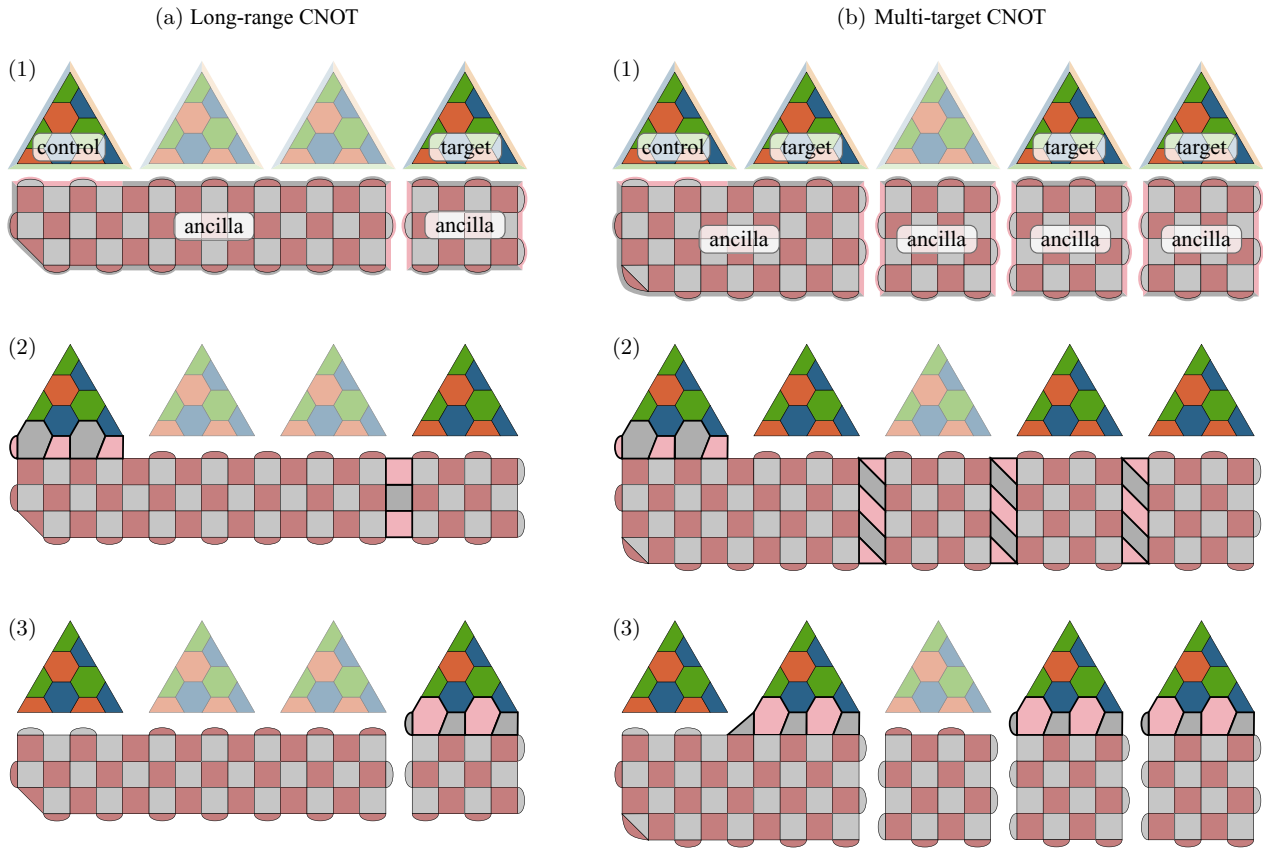


FIG. 6. (a) By simultaneously measuring the ZZ parities between the control and two ancillas (a2), one can use a long ancilla qubit for long-range CNOTs. The unused long ancilla is read out in the σ_x basis. (b) The same protocol can be used for long-range multitarget CNOTs, where multiple ZZ- and XX-parity measurements (b2),(b3) are carried out simultaneously. All protocols have a space overhead that scales with $O(s \ln s)$ of the control-target separation s , and a time overhead that scales with $O(\ln s)$.

maintain the same CNOT accuracy. With a linearly growing length of the surface code qubits, the space overhead of the lattice surgery CNOT protocol is $O(s \ln s)$. For the time overhead, one needs to take the classical overhead of decoding and the effect of measurement errors during syndrome readout into account. There exist surface code decoders with a runtime of $O(\ln s)$ [49]. The correction of measurement errors requires recording multiple rounds of syndrome extraction for one code cycle, depending on the measurement fidelity [50], effectively extending the code into a third time dimension. These “time errors” are suppressed exponentially with the number of measurement rounds, but the number of possible error strings increases linearly with s . Thus, similar to the space overhead, measurement errors increase the time overhead by $O(\ln s)$, and the total time overhead is still only $O(\ln s)$.

Note that the code distances (given by the width) of the ancilla qubits need not be as high as the code distance of the color-code qubits, since the ancillas only need to survive for the few code cycles of the CNOT protocol, as opposed to data qubits that may need to survive for the entire quantum computation. In our example, the ancilla qubits have distances $d = 3$, $d = 4$, and $d = 5$ in the protocols in Figs. 4 and 6. However, we expect that for most practical quantum computations, the entire space allocated for CNOT ancillas will be in use for different CNOTs essentially for the entire

duration of the computation. Thus, for most practical purposes, the width of the ancilla qubits and the code distance of the color-code qubits can be chosen to be equal [as in Fig. 6(b)], and the logarithmic scaling of the ancilla width can be ignored. With this approach, all parts of the code are protected against error strings of length $(d - 1)/2$ during each code cycle. The logarithmic scaling merely implies that for a quantum computation involving n logical qubits, the necessary code distance to reach a target error probability at the end of the quantum computation scales with $O(\ln n)$. Again, we point out that by identifying two copies of surface code qubits as one color-code qubit [51], this CNOT protocol can be done entirely using color codes, as we show in Appendix B. However, this uses more physical qubits than the surface code approach, and requires the measurement of eight-qubit stabilizers.

Both surface and color codes can be implemented on the square lattice of tetrons shown in Fig. 2, since lattice surgery only requires the measurement of additional stabilizers, i.e., the measurement of 4-, 8-, and 12-Majorana operators. While our examples have illustrated logical qubits arranged on a line, this protocol can be straightforwardly extended to two-dimensional arrangements of logical qubits. One possible 2D arrangement of color-code qubits is shown in Fig. 7, where qubits are arranged in blocks of six. The figure also shows two surface code ancilla qubits that can be used for a CNOT between

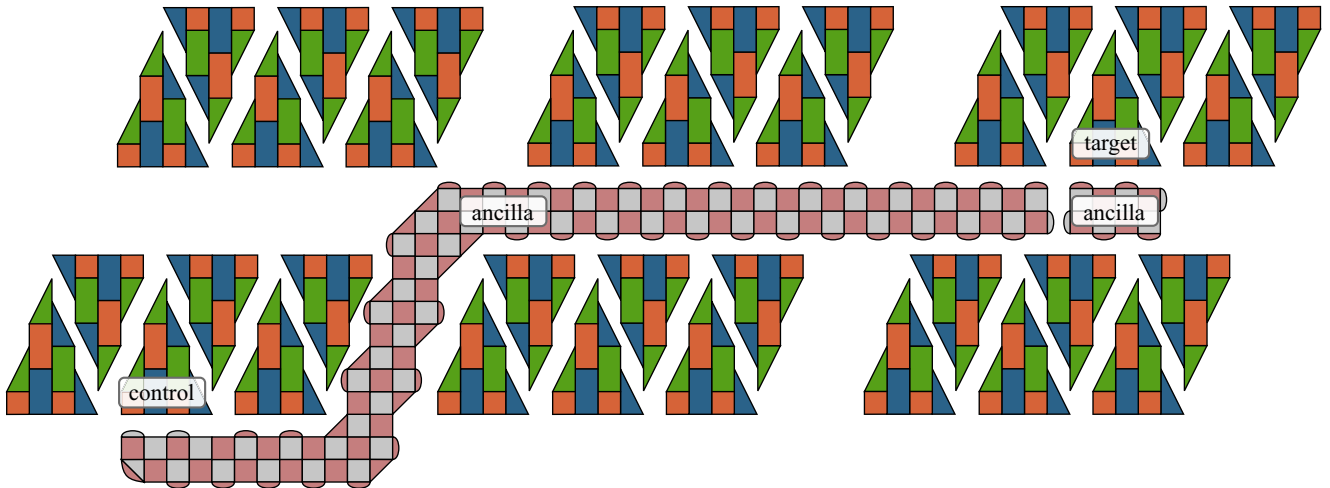


FIG. 7. Example of a two-dimensional arrangement of color-code qubits on a square lattice, and an example of surface code ancillas used for a long-range CNOT between distant qubits. For the next CNOT gate, these ancillas are discarded and the space between qubit blocks can be used to initialize different ancillas. The separation between blocks of color-code qubits dictates the maximum code distance of the surface code ancillas and influences the number of CNOT gates that can be performed in parallel. With larger separation, multiple “lanes” of ancilla qubits can fit between blocks, allowing for multiple overlapping multitarget CNOT gates.

distant color-code qubits. In this way, lattice surgery can provide long-range communication between any two logical qubits with essentially constant time overhead.

V. CONCLUSION

Current ideas for realizing a Majorana-based quantum computer rely on nanowire arrays such as networks of tetrons that only allow for local Majorana parity operator measurements and physical T gates. Here, we have shown how Majorana-based qubits can be combined with color codes for universal fault-tolerant quantum computation without the need for moving Majoranas. In our architecture, logical single-qubit Clifford gates are implemented by Majorana tracking, which minimizes their overhead. Furthermore, we combine surface codes with color codes using surface-to-color-code lattice surgery, which yields long-range multitarget CNOT gates with a time overhead that scales only with $O(\ln s)$ of the distance s between the control and target qubits and a space overhead that scales with $O(s \ln s)$. Moreover, this approach features a

lower space overhead and lower-weight stabilizers compared to a purely color-code-based scheme.

Logical T gates are the most expensive operation in this scheme, as they require magic state distillation. Their overhead can be reduced by improving the fidelity of physical T gates, and by exploring faster distillation protocols and alternatives to magic state distillation. As to the concrete physical implementation, there are several proposals for architectures that implement the two operations required of nanowire arrays. Still, none of these architectures are particularly optimized towards error correction with color codes. Optimizing for fast stabilizer measurement, high measurement fidelity, and low physical error rate is crucial to ensure scalability. Exploring efficient decoding schemes for color and surface code qubits can further reduce the classical overhead.

What is more, our scheme can also be applied to nontopological architectures, such as superconducting qubits, albeit without the advantages of robust physical single-qubit Clifford gates and ancilla-free syndrome readout. For this reason, these architectures usually favor surface codes over color codes due to the easier four-qubit stabilizer measurements of

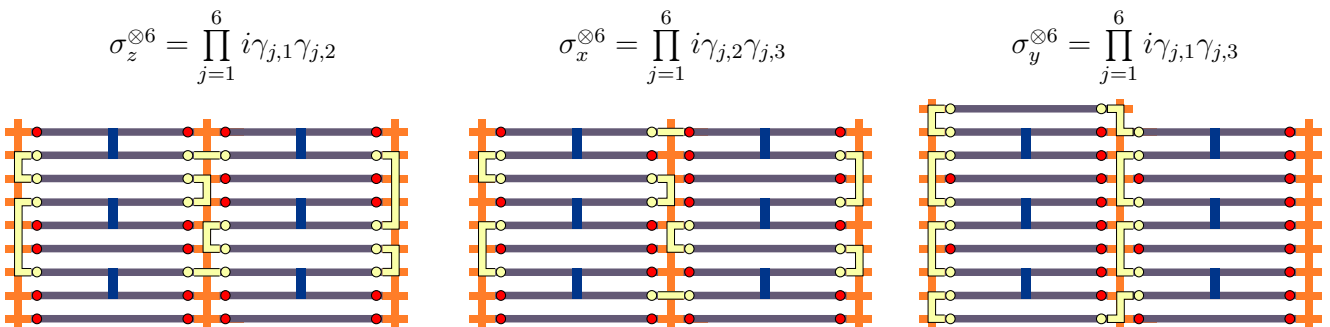


FIG. 8. Tunnel coupling configurations for color-code stabilizer measurements with tetrons that feature only one vertical semiconductor wire between tetrons, as in the architecture of Ref. [41].

surface codes compared to the weight-six stabilizers of color codes. Even though surface codes do not feature transversal single-qubit Clifford gates, they can still be used to implement single-qubit Clifford gates with zero time overhead [43,52,53] by encoding logical qubits in surface code twist defects [54]. Still, surface code qubits in this scheme suffer from a lack of easy σ_y measurements, and require $\sim d^2$ physical qubits for each logical qubit, while the color-code approach discussed in this work only requires $\sim \frac{3}{4}d^2$ physical qubits to achieve the same code distance d .

We note that the twist-based triangle codes presented in Ref. [52] also feature a space overhead of $\sim \frac{3}{4}d^2$ and manage to implement easy σ_y measurements, but they encode the logical σ_x , σ_y , and σ_z information in each of the three sides of the triangles separately. Therefore, these qubits cannot be packed as densely as the color-code qubits in Fig. 7, since all three sides of each triangle need to be accessible by lattice surgery. This either implies an increased space overhead by requiring some free space as padding around the triangles, or it introduces a time overhead for logical single-qubit Clifford gates by requiring code operations for the reorientation of triangle qubits.

The spatial overhead of the color-code scheme, on the other hand, can be reduced even further to $\sim \frac{1}{2}d^2$ by using 4.8.8 color codes [28] instead of the 6.6.6 color codes discussed in this work. However, this comes at the price of a higher-weight stabilizer, as 4.8.8 color codes feature eight-qubit stabilizers, instead of just six-qubit stabilizers. If higher-weight stabilizers are not significantly more difficult to measure, which could hold true for Majorana-based qubits, it is advantageous to use the color-code-based scheme instead of a pure surface code architecture in order to reduce the overhead of fault-tolerant quantum computing.

ACKNOWLEDGMENTS

We thank C. E. Bardyn, J. Eisert, T. Karzig, M. S. Kesselring, and S. Plugge for illuminating discussions. This work has been supported by the Deutsche Forschungsgemeinschaft (Bonn) within the network CRC TR 183.

APPENDIX A: STABILIZER MEASUREMENTS WITH SINGLE-WIRE TETRONS

Here, we show how the measurement of the color-code stabilizers shown in Fig. 2 can be implemented in a network of tetrans that features only one vertical semiconductor wire between tetrans, which is the architecture discussed in Ref. [41]. The corresponding tunnel coupling configurations in this architecture are shown in Fig. 8.

In the configuration of Fig. 2 with two vertical wires, all stabilizers of the same color can be measured simultaneously. This is no longer the case in Fig. 8, as these measurements

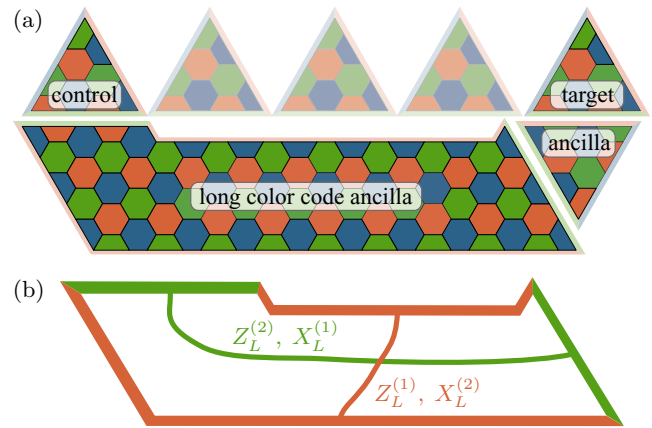


FIG. 9. (a) Qubit arrangement for long-range CNOTs by lattice surgery using a color-code ancilla. (b) The long color-code qubit used in this protocol actually encodes two logical qubits, as it has two red boundaries and two green boundaries. Red-to-red strings define the logical operators $Z_L^{(1)}$ and $X_L^{(2)}$, while green-to-green strings are the operators $Z_L^{(2)}$ and $X_L^{(1)}$.

use a vertical wire or a coherent link of a neighboring six-qubit block. In particular, the $\sigma_z^{\otimes 6}$ and $\sigma_x^{\otimes 6}$ measurements overlap with their left (or right) neighbors, whereas the $\sigma_y^{\otimes 6}$ measurement overlaps with the upper neighbor. Therefore, syndrome extraction requires *two* measurement rounds for the measurement of each stabilizer type, as opposed to just one.

APPENDIX B: LATTICE SURGERY WITH COLOR-CODE ANCILLAS

The long-range CNOT protocol can also be done using color-code ancillas. Here, the long surface code qubit is replaced by a long color-code qubit; see Fig. 9(a). This is a color-code qubit with two red and two green boundaries, as shown in Fig. 9(b). It is defined by n physical qubits and $n - 2$ stabilizers, and therefore encodes two logical qubits. However, since the code distance of such a qubit is always even, the support of the X_L and Z_L operators of the individual logical qubits cannot coincide. Instead, strings of Pauli operators that connect two green boundaries encode the operators $X_L^{(1)}$ and $Z_L^{(2)}$, and red-to-red strings are $X_L^{(2)}$ and $Z_L^{(1)}$, where the superscript labels the logical qubit.

In this way, long color-code qubits are equivalent to two surface code qubits on top of each other with a relative rotation of 90° . Therefore, the long-range CNOT protocol is the same as for surface code ancillas, but only uses one of the two encoded logical qubits. This redundancy is also manifested in a higher number of physical qubits than in surface code ancillas. Moreover, lattice surgery between color codes requires eight-qubit stabilizer measurements.

[1] A. Kitaev, Fault-tolerant quantum computation by anyons, *Ann. Phys. (NY)* **303**, 2 (2003).
 [2] C. Nayak, S. H. Simon, A. Stern, M. Freedman, and S. D. Sarma, Non-Abelian anyons and topological quantum computation, *Rev. Mod. Phys.* **80**, 1083 (2008).

[3] B. M. Terhal, Quantum error correction for quantum memories, *Rev. Mod. Phys.* **87**, 307 (2015).
 [4] L. Fu and C. L. Kane, Superconducting Proximity Effect and Majorana Fermions at the Surface of a Topological Insulator, *Phys. Rev. Lett.* **100**, 096407 (2008).

- [5] Y. Oreg, G. Refael, and F. von Oppen, Helical Liquids and Majorana Bound States in Quantum Wires, *Phys. Rev. Lett.* **105**, 177002 (2010).
- [6] R. M. Lutchyn, J. D. Sau, and S. D. Sarma, Majorana Fermions and a Topological Phase Transition in Semiconductor-Superconductor Heterostructures, *Phys. Rev. Lett.* **105**, 077001 (2010).
- [7] J. Alicea, New directions in the pursuit of Majorana fermions in solid state systems, *Rep. Prog. Phys.* **75**, 076501 (2012).
- [8] C. W. J. Beenakker, Search for Majorana fermions in superconductors, *Annu. Rev. Condens. Matter Phys.* **4**, 113 (2013).
- [9] V. Mourik, K. Zuo, S. M. Frolov, S. R. Plissard, E. P. a. M. Bakkers, and L. P. Kouwenhoven, Signatures of Majorana fermions in hybrid superconductor-semiconductor nanowire devices, *Science* **336**, 1003 (2012).
- [10] A. Das, Y. Ronen, Y. Most, Y. Oreg, M. Heiblum, and H. Shtrikman, Zero-bias peaks and splitting in an Al-InAs nanowire topological superconductor as a signature of Majorana fermions, *Nat. Phys.* **8**, 887 (2012).
- [11] S. M. Albrecht, A. P. Higginbotham, M. Madsen, F. Kuemmeth, T. S. Jespersen, J. Nyg, P. Krogstrup, and C. M. Marcus, Exponential protection of zero modes in Majorana islands, *Nature (London)* **531**, 206 (2016).
- [12] M. Deng, S. Vaitiekėnas, E. Hansen, J. Danon, M. Leijnse, K. Flensberg, J. Nygård, P. Krogstrup, and C. Marcus, Majorana bound state in a coupled quantum-dot hybrid-nanowire system, *Science* **354**, 1557 (2016).
- [13] H. J. Suominen, M. Kjaergaard, A. R. Hamilton, J. Shabani, C. J. Palmström, C. M. Marcus, and F. Nichele, Zero-Energy Modes from Coalescing Andreev States in a Two-Dimensional Semiconductor-Superconductor Hybrid Platform, *Phys. Rev. Lett.* **119**, 176805 (2017).
- [14] D. J. van Woerkom, A. Geresdi, and L. P. Kouwenhoven, One minute parity lifetime of a NbTiN Cooper-pair transistor, *Nat. Phys.* **11**, 547 (2015).
- [15] A. P. Higginbotham, S. M. Albrecht, G. Kiršanskas, W. Chang, F. Kuemmeth, P. Krogstrup, T. S. Jespersen, J. Nygard, K. Flensberg, and C. M. Marcus, Parity lifetime of bound states in a proximitized semiconductor nanowire, *Nat. Phys.* **11**, 1017 (2015).
- [16] S. M. Albrecht, E. B. Hansen, A. P. Higginbotham, F. Kuemmeth, T. S. Jespersen, J. Nygård, P. Krogstrup, J. Danon, K. Flensberg, and C. M. Marcus, Transport Signatures of Quasiparticle Poisoning in a Majorana Island, *Phys. Rev. Lett.* **118**, 137701 (2017).
- [17] D. Rainis and D. Loss, Majorana qubit decoherence by quasiparticle poisoning, *Phys. Rev. B* **85**, 174533 (2012).
- [18] J. Preskill, Reliable quantum computers, *Proc. R. Soc. London A* **454**, 385 (1998).
- [19] S. Bravyi, B. M. Terhal, and B. Leemhuis, Majorana fermion codes, *New J. Phys.* **12**, 083039 (2010).
- [20] S. Vijay, T. H. Hsieh, and L. Fu, Majorana Fermion Surface Code for Universal Quantum Computation, *Phys. Rev. X* **5**, 041038 (2015).
- [21] S. Vijay and L. Fu, Physical implementation of a Majorana fermion surface code for fault-tolerant quantum computation, *Phys. Scr.* **T168**, 014002 (2016).
- [22] L. A. Landau, S. Plugge, E. Sela, A. Altland, S. M. Albrecht, and R. Egger, Towards Realistic Implementations of a Majorana Surface Code, *Phys. Rev. Lett.* **116**, 050501 (2016).
- [23] S. Plugge, L. A. Landau, E. Sela, A. Altland, K. Flensberg, and R. Egger, Roadmap to Majorana surface codes, *Phys. Rev. B* **94**, 174514 (2016).
- [24] S. Vijay and L. Fu, Quantum error correction for complex and Majorana fermion qubits, [arXiv:1703.00459](https://arxiv.org/abs/1703.00459).
- [25] M. B. Hastings, Small Majorana fermion codes, [arXiv:1703.00612](https://arxiv.org/abs/1703.00612).
- [26] D. Litinski, M. S. Kesselring, J. Eisert, and F. von Oppen, Combining Topological Hardware and Topological Software: Color-Code Quantum Computing with Topological Superconductor Networks, *Phys. Rev. X* **7**, 031048 (2017).
- [27] H. Bombin and M. A. Martin-Delgado, Topological Quantum Distillation, *Phys. Rev. Lett.* **97**, 180501 (2006).
- [28] A. J. Landahl, J. T. Anderson, and P. R. Rice, Fault-tolerant quantum computing with color codes, [arXiv:1108.5738](https://arxiv.org/abs/1108.5738).
- [29] 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).
- [30] E. T. Campbell, B. M. Terhal, and C. Vuillot, Roads towards fault-tolerant universal quantum computation, *Nature (London)* **549**, 172 (2017).
- [31] C. Horsman, A. G. Fowler, S. Devitt, and R. V. Meter, Surface code quantum computing by lattice surgery, *New J. Phys.* **14**, 123011 (2012).
- [32] A. J. Landahl and C. Ryan-Anderson, Quantum computing by color-code lattice surgery, [arXiv:1407.5103](https://arxiv.org/abs/1407.5103).
- [33] S. Bravyi and A. Kitaev, Universal quantum computation with ideal Clifford gates and noisy ancillas, *Phys. Rev. A* **71**, 022316 (2005).
- [34] J. Alicea, Y. Oreg, G. Refael, F. von Oppen, and M. P. A. Fisher, Non-Abelian statistics and topological quantum information processing in 1D wire networks, *Nat. Phys.* **7**, 412 (2011).
- [35] B. Van Heck, A. Akhmerov, F. Hassler, M. Burrello, and C. Beenakker, Coulomb-assisted braiding of Majorana fermions in a Josephson junction array, *New J. Phys.* **14**, 035019 (2012).
- [36] T. Hyart, B. van Heck, I. C. Fulga, M. Burrello, A. R. Akhmerov, and C. W. J. Beenakker, Flux-controlled quantum computation with Majorana fermions, *Phys. Rev. B* **88**, 035121 (2013).
- [37] D. Aasen, M. Hell, R. V. Mishmash, A. Higginbotham, J. Danon, M. Leijnse, T. S. Jespersen, J. A. Folk, C. M. Marcus, K. Flensberg, and J. Alicea, Milestones Toward Majorana-Based Quantum Computing, *Phys. Rev. X* **6**, 031016 (2016).
- [38] F. L. Pedrocchi and D. P. DiVincenzo, Majorana Braiding with Thermal Noise, *Phys. Rev. Lett.* **115**, 120402 (2015).
- [39] S. Plugge, A. Rasmussen, R. Egger, and K. Flensberg, Majorana box qubits, *New J. Phys.* **19**, 012001 (2017).
- [40] S. Vijay and L. Fu, Teleportation-based quantum information processing with Majorana zero modes, *Phys. Rev. B* **94**, 235446 (2016).
- [41] T. Karzig, C. Knapp, R. M. Lutchyn, P. Bonderson, M. B. Hastings, C. Nayak, J. Alicea, K. Flensberg, S. Plugge, Y. Oreg, C. M. Marcus, and M. H. Freedman, Scalable designs for quasiparticle-poisoning-protected topological quantum computation with Majorana zero modes, *Phys. Rev. B* **95**, 235305 (2017).
- [42] P. Bonderson, M. Freedman, and C. Nayak, Measurement-Only Topological Quantum Computation, *Phys. Rev. Lett.* **101**, 010501 (2008).

- [43] M. B. Hastings and A. Geller, Reduced space-time and time costs using dislocation codes and arbitrary ancillas, *Quantum Info. Comput.* **15**, 962 (2015).
- [44] D. J. Clarke, J. D. Sau, and S. D. Sarma, A Practical Phase Gate for Producing Bell Violations in Majorana Wires, *Phys. Rev. X* **6**, 021005 (2016).
- [45] H. P. Nautrup, N. Friis, and H. J. Briegel, Fault-tolerant interface between quantum memories and quantum processors, [arXiv:1609.08062](https://arxiv.org/abs/1609.08062).
- [46] A. Paler, S. Devitt, K. Nemoto, and I. Polian, Software-based Pauli tracking in fault-tolerant quantum circuits, in *Design, Automation and Test in Europe Conference and Exhibition* (IEEE, New York, 2014), pp. 1–4.
- [47] D. Gottesman, The Heisenberg representation of quantum computers, *Proc. XXII Int. Coll. Group. Th. Meth. Phys.* **1**, 32 (1999).
- [48] T. Karzig, Y. Oreg, G. Refael, and M. H. Freedman, Universal Geometric Path to a Robust Majorana Magic Gate, *Phys. Rev. X* **6**, 031019 (2016).
- [49] G. Duclos-Cianci and D. Poulin, Fast Decoders for Topological Quantum Codes, *Phys. Rev. Lett.* **104**, 050504 (2010).
- [50] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Topological quantum memory, *J. Math. Phys.* **43**, 4452 (2002).
- [51] A. Kubica, B. Yoshida, and F. Pastawski, Unfolding the color code, *New J. Phys.* **17**, 083026 (2015).
- [52] T. J. Yoder and I. H. Kim, The surface code with a twist, *Quantum* **1**, 2 (2017).
- [53] D. Litinski and F. von Oppen, Lattice surgery with a twist: Simplifying Clifford gates of surface codes, [arXiv:1709.02318](https://arxiv.org/abs/1709.02318).
- [54] H. Bombin, Topological Order with a Twist: Ising Anyons from an Abelian Model, *Phys. Rev. Lett.* **105**, 030403 (2010).

5 | Color Codes in Braiding-Based Networks

So far, our Majorana-based implementations have focused on qubits based on the tetrons and hexons introduced by Karzig et al. in Ref. [13]. In the following publication, we implement color codes in networks of topological superconductors inspired by the T-junction-based constructions of Aasen et al. in Ref. [14].

Combining Topological Hardware and Topological Software: Color-Code Quantum Computing with Topological Superconductor Networks

Daniel Litinski, Markus S. Kesselring, Jens Eisert, and Felix von Oppen

*Dahlem Center for Complex Quantum Systems and Fachbereich Physik,
Freie Universität Berlin, Arnimallee 14, 14195 Berlin, Germany*

(Received 12 April 2017; revised manuscript received 4 July 2017; published 15 September 2017)

We present a scalable architecture for fault-tolerant topological quantum computation using networks of voltage-controlled Majorana Cooper pair boxes and topological color codes for error correction. Color codes have a set of transversal gates which coincides with the set of topologically protected gates in Majorana-based systems, namely, the Clifford gates. In this way, we establish color codes as providing a natural setting in which advantages offered by topological hardware can be combined with those arising from topological error-correcting software for full-fledged fault-tolerant quantum computing. We provide a complete description of our architecture, including the underlying physical ingredients. We start by showing that in topological superconductor networks, hexagonal cells can be employed to serve as physical qubits for universal quantum computation, and we present protocols for realizing topologically protected Clifford gates. These hexagonal-cell qubits allow for a direct implementation of open-boundary color codes with ancilla-free syndrome read-out and logical T gates via magic-state distillation. For concreteness, we describe how the necessary operations can be implemented using networks of Majorana Cooper pair boxes, and we give a feasibility estimate for error correction in this architecture. Our approach is motivated by nanowire-based networks of topological superconductors, but it could also be realized in alternative settings such as quantum-Hall–superconductor hybrids.

DOI: [10.1103/PhysRevX.7.031048](https://doi.org/10.1103/PhysRevX.7.031048)

Subject Areas: Condensed Matter Physics,
Quantum Information

I. INTRODUCTION

Physical realizations of large-scale quantum computers remain a paramount experimental challenge because of the unavoidable presence of environmental decoherence. Topological quantum computing is generally seen as paving the way towards a solution to this problem [1–3] in more than one sense: In the mindset of condensed-matter physics, excitations of topological phases of matter have been identified as candidates for physical qubits that are robust to local perturbations and on which a certain set of quantum gate operations can be performed largely noise-free. In the context of quantum information theory, topological quantum error-correcting codes have been devised as codes featuring high error tolerance which only require the measurement of local stabilizer operators. While clearly related, these predominantly hardware-based and software-based approaches constitute two distinctly different readings of topological quantum computing.

On the hardware side, the interplay of superconductivity, spin-orbit coupling, and single spin-polarized conducting

channels has inspired various proposals for experimental realizations of Majorana zero modes [4–9], subsequently simply referred to as Majoranas. Quantum information can be encoded using spatially separated pairs of Majoranas [10] whose parity state is unaffected by local perturbations. We refer to qubits encoded using this parity state as physical qubits arising from topological hardware. Furthermore, the exchange of pairs of Majoranas constitutes a nontrivial braiding operation that can be used for the implementation of robust quantum gates. Recent experiments have provided increasing evidence for the emergence of Majorana zero modes in semiconducting nanowires with mesoscopic superconducting islands [11–15]. In such setups, the state of the Majorana pair depends on the fermion parity of the mesoscopic island. Therefore, electrons tunneling onto the island can change the parity state and thus spoil any quantum information encoded by the Majoranas. This process is called quasiparticle poisoning. Among other error sources, its rate defines a finite lifetime for Majorana-based qubits.

If one aims at storing and manipulating quantum information beyond the quasiparticle poisoning time—in principle, for arbitrary times—errors need to be actively corrected. This can be achieved by making use of topological error-correcting codes. The basic principle of such codes is to fight local errors with entanglement so that local noise cannot affect the logical information [16]. This is

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

done by using multiple physical qubits to encode a single logical qubit, which we refer to as topological software.

Recent proposals [17–21] have taken key steps in the direction of combining topological hardware with software. Importantly, the combination of Majorana-based qubits with topological surface codes has been studied. However, while the replacement of physical qubits with logical qubits enhances resilience against noise, one must be aware that physical gates are also substituted with logical gates. Topological protection of gates on the physical level does not necessarily translate to logical gates since, in general, physical and logical gates are unrelated. Any error-correcting code is, in principle, allowed to have a (nonuniversal [22]) set of transversal gates—i.e., logical gates that correspond to the simultaneous application of the same physical gate to all physical qubits [23]—but the transversal gates of surface codes are limited to the CNOT and Pauli gates.

In this work, we go a significant step further and establish two-dimensional topological color codes as a natural fit to enhance the fault tolerance of Majorana-based quantum computers. They seamlessly combine the fermion-parity-protected topological order of topological superconductors [24] with the long-range topological order of the toric code [25] in a way that allows one to exploit the topological protection of both. Compared to surface codes, color codes not only have a richer set of transversal gates, but this set also coincides with the gates that are accessible by braiding of Majoranas, namely, the Clifford gates [26,27]. We hence further contribute to identifying the precise advantages offered by topological protection, both as far as the underlying condensed-matter physics is concerned and on the level of logical encoding.

In the following sections, we describe our design for a scalable fault-tolerant topological quantum computer from

the ground up, discussing the microscopic details of the Majorana-based physical qubits, their encoding in topological superconductor networks, and the arrangement and manipulation of logical qubits for quantum computing (see Fig. 1). We begin in Sec. II by describing how networks of topological superconducting islands can be used for universal quantum computation with topologically protected Clifford gates. We require that topological superconductor networks are capable of three operations: moving Majoranas through the network by coupling neighboring islands, measuring $2n$ -Majorana parity operators on connected islands, and lifting the degeneracy of the parity states on an island. We show that in such networks, physical qubits can be arranged in hexagonal cells with six nearest neighbors such that the qubits form a triangular lattice [gray hexagons in Fig. 1(b)]. Here, each hexagonal cell is associated with four Majoranas that are used for quantum computation. Universal quantum computation requires the implementation of a universal set of quantum gates. One such set consists of the Clifford gates (Hadamard, $\pi/4$, and CNOT gates) and the T gate (or $\pi/8$ gate). We present protocols for single-qubit Clifford gates via braiding inside a hexagonal cell and CNOT gates between any pair of cells via braiding and parity measurements. The addition of an unprotected T gate—which is not accessible via braiding of Majoranas—by controlled splitting of the degeneracy completes the universal gate set.

While the Clifford gates of these Majorana-based qubits are topologically protected, the T gate requires fine-tuning of the device control parameters, which can easily lead to errors in the T gate. Instead of attempting to implement a robust T gate on the level of physical qubits [28,29], we address this problem using magic-state distillation. This is a common proposal for a fault-tolerant implementation of the T gate on the level of logical qubits, the precision of which

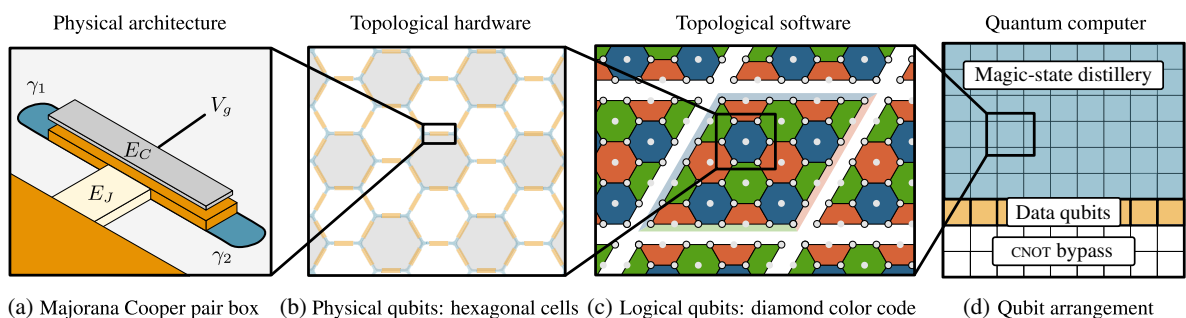


FIG. 1. Overview of the design for a scalable fault-tolerant topological quantum computer. The basic building block is the Majorana Cooper pair box (a) consisting of a topological superconducting island with charging energy E_C and Josephson energy E_J hosting a pair of Majoranas γ_1 and γ_2 . Parity measurements of the island are controlled by a gate voltage V_g . Multiple connected Majorana Cooper pair boxes form a topological superconductor network through which Majoranas can be moved and which allows for the measurement of $2n$ -Majorana parity operators. A triangular lattice of hexagonal-cell qubits (b) allows for universal quantum computation with topologically protected Clifford gates. Fault tolerance is added by encoding hexagonal-cell qubits in diamond color codes (c) with transversal Clifford gates. These form a square lattice of logical qubits. Arranging qubits on a line (d) with a magic-state distillery and a CNOT bypass completes the universal gate set with a logical T gate and allows for CNOT gates between any pair of data qubits with constant-time overhead.

scales with the protocol length [30]. These protocols typically include many multitarget CNOT gates (i.e., multiple CNOT gates with the same control but different target qubits). We show how parity measurements in topological superconductor networks can be used for fast multitarget CNOT gates, replacing multiple CNOT gates by a protocol that is as fast as a single CNOT.

Another advantage of Majorana-based qubits is ancilla-free syndrome read-out. Quantum error-correcting codes typically require the measurement of stabilizer operators of the form $\sigma_z^{\otimes n}$, where σ_z is a Pauli matrix and n is the number of qubits involved in the measurement. In conventional setups for quantum computing, such n -qubit parity operators are typically not directly measurable but require a lengthy protocol involving an ancilla qubit and n CNOT gates. Since in topological superconductor networks the parity of $2n$ Majoranas can be measured directly if the Majoranas are moved onto a single connected superconducting island, n -qubit parity operators can be measured without the use of ancilla qubits. In preparation for the color code, we demonstrate how hexagonal-cell qubits can be used to measure the required six-qubit parity operators.

The triangular lattice of physical qubits allows for a direct implementation of triangular color codes with transversal Clifford gates. In contrast to fermionic codes [18,31–33] where each lattice site corresponds to a Majorana fermion, we use a bosonic code where each lattice site is a bosonic degree of freedom (d.o.f.) since our physical qubits are comprised of *four* Majorana fermions each and are therefore bosonic qubits. In our encoding scheme, the logical qubits are arranged on a square lattice, where each logical qubit has four nearest neighbors. As this leaves some unused hexagonal cells, we extend the triangular color codes to diamond-shaped color codes [see Fig. 1(c)], which have the same code distance as their triangular counterparts but a lower logical error rate. In Sec. III, we show that a square arrangement of diamond color-code qubits [see Fig. 1(d)] can be used for universal fault-tolerant quantum computing with topologically protected Clifford gates, constant-time CNOT gates between any pair of logical qubits, and logical T gates with arbitrary precision. We discuss various protocols for logical CNOT and multitarget CNOT gates, based on transversal gates and lattice surgery [34].

In order to show a possible scalable realization of topological superconductor networks, we review Majorana Cooper pair boxes [18,35–40] in Sec. IV as basic building blocks of the physical architecture. In our description of Majorana Cooper pair boxes [see Fig. 1(a)], we revisit how topological superconducting islands combined with capacitive coupling via a top gate and Josephson coupling to a bulk superconductor can be used for parity-to-charge conversion [35]. We demonstrate that networks of Majorana Cooper pair boxes are capable of performing the aforementioned required operations. These can be implemented using proximitized semiconductor nanowires,

on which recent experiments have focused, but possibly also in other platforms such as hybrid structures based on quantum Hall, quantum spin Hall, or quantum anomalous Hall edge states.

Finally, in Sec. V, we consider the main error sources in our physical architecture and give a feasibility estimate. There are three time scales that characterize networks of Majorana Cooper pair boxes: the time required to move Majoranas, the duration of parity measurements, and the quasiparticle poisoning time. We identify constraints that physical setups need to satisfy in order to operate below the error threshold of color codes. Using a Monte Carlo simulation, we study the improved performance of diamond color codes over triangular color codes and give an estimate of the space overhead—i.e., the number of physical qubits per logical qubit—required for the logical qubits to reach sufficiently long survival times for quantum computation on the basis of experimental measurements of quasiparticle poisoning times [41–43].

It should be clear that this article is aimed at both the condensed-matter and quantum information communities. Therefore, we have made an effort to include basic introductions to the relevant concepts. Still, this article is by no means a review, but it is meant to lay the groundwork for color-code quantum computing with Majoranas in order to fully exploit the topological protection of Majorana-based qubits.

II. TOPOLOGICAL HARDWARE: HEXAGONAL-CELL QUBITS

In a topological superconductor network, each superconducting island can host a pair of Majoranas γ_1 and γ_2 with degenerate even $|e\rangle$ and odd $|o\rangle$ eigenstates of the fermion-parity operator $i\gamma_1\gamma_2$. We require the network to be capable of three basic operations:

- (1) Majoranas can be moved from island to island by connecting neighboring superconducting islands (see Fig. 2).
- (2) For $2n$ Majoranas on a single connected island, the total parity operator $i^n \prod_{j=1}^{2n} \gamma_j$ of $2n$ Majoranas can be measured projectively.
- (3) The degeneracy between $|e\rangle$ and $|o\rangle$ can be split temporarily and restored again.

We now show that such networks can be used to realize a universal quantum computer. Even though a pair of Majoranas is a two-level system, no superposition of $|e\rangle$ and $|o\rangle$ can exist because of fermion-parity superselection, and therefore, a pair of Majoranas cannot be used as a qubit. Instead, qubits are encoded using two islands hosting four Majoranas with fixed *total* fermion parity (see Fig. 2), either in the even-parity sector,

$$|0\rangle = |e, e\rangle, \quad |1\rangle = |o, o\rangle, \quad (1)$$

or in the odd-parity sector,

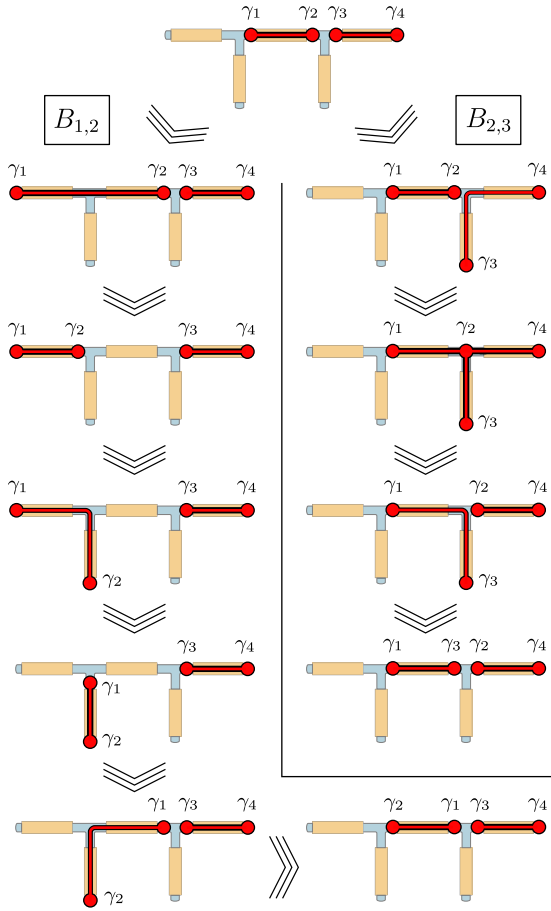


FIG. 2. Protocols for braiding operations in a double T junction, where red dots denote Majoranas and red lines connect the coupled superconducting islands (orange). Left diagrams: Braiding of γ_1 and γ_2 is achieved via a three-point turn in the left T junction. Right diagrams: To braid γ_2 and γ_3 , first γ_3 is moved from the right island to the bottom right island. Then, γ_2 is moved to the right island by first connecting all three islands in the right T junction and then disconnecting the right island. Finally, γ_3 is moved to the center island.

$$|0\rangle = |e, o\rangle, \quad |1\rangle = |o, e\rangle. \quad (2)$$

To initialize a qubit in one of these states, the two-Majorana fermion parity of both islands is measured. Both encodings can be used interchangeably, as in both cases, the qubit is measured in the computational basis by measuring the parity on the first island.

Furthermore, in both encodings, the exchange of γ_1 and γ_2 , and of γ_2 and γ_3 performs the same braiding operations $B_{1,2}$ and $B_{2,3}$, respectively. Since the braiding operator [8]

$$B_{i,j} = \frac{1 + \gamma_i \gamma_j}{\sqrt{2}} \quad (3)$$

describes the clockwise exchange of Majoranas γ_i and γ_j , the braiding operators describe the qubit operations

$$B_{1,2} = e^{-i(\pi/4)\sigma_z}, \quad B_{2,3} = e^{-i(\pi/4)\sigma_x}. \quad (4)$$

Here, σ_z and σ_x are Pauli operators in the computational basis $\{|0\rangle, |1\rangle\}$. In terms of Majorana operators, $\sigma_z = i\gamma_1\gamma_2$ and $\sigma_x = i\gamma_2\gamma_3$.

Universal quantum computation requires a universal set of quantum gates, i.e., a set of unitary operations on the qubits, such that any n -qubit unitary operation can be constructed as a product of unitaries from the universal set. One such universal gate set is the standard set $\{H, T, S, \text{CNOT}\}$ [44], in which $S = \exp(-i\pi\sigma_z/4)$ and $T = \exp(-i\pi\sigma_z/8)$ are the S and T gates (equivalently $\pi/4$ and $\pi/8$ gates), and H and CNOT are the Hadamard and controlled-NOT gate,

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad \text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \quad (5)$$

The gates generated by the nonuniversal set $\{H, S, \text{CNOT}\}$ form the set of so-called Clifford gates, which are those gates that map multiqubit Pauli operators to Pauli operators under conjugation.

A. Clifford gates in hexagonal-cell qubits

From Eq. (4), it is evident that the single-qubit Clifford gates can be implemented by braiding since $S = B_{1,2}$ and $H = iB_{1,2}B_{2,3}B_{1,2} = iB_{2,3}B_{1,2}B_{2,3}$. A topological superconductor network that allows for both exchanges is the double T junction [35,45]. In this five-island geometry, the upper and right superconducting islands host four Majoranas encoding a qubit. Figure 2 shows protocols for the braiding operations $B_{1,2}$ via a three-point turn in the left T junction and for $B_{2,3}$ using the right T junction.

In the remainder of this section, we show that arrays of hexagonal-cell qubits depicted in Fig. 3 can be used for universal quantum computation, where qubits are arranged

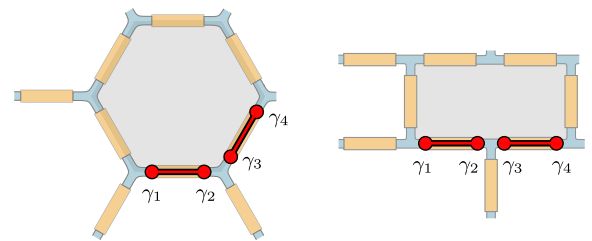


FIG. 3. Left diagram: Hexagonal cell hosting four Majoranas encoding one qubit. The single-qubit Clifford gates can be performed by braiding in the double T junction in the lower part of the cell. In a network of such cells, each cell has up to six neighbors. Right diagram: Such a hexagonal lattice can also be realized with only two different wire orientations using a brick-wall geometry of superconducting islands.

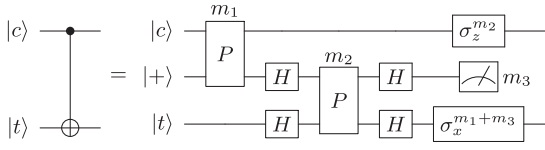


FIG. 4. Quantum circuit for a CNOT gate using parity measurements and an ancilla qubit initialized in the $|+\rangle$ state. First, the parity operator $\sigma_z \otimes \sigma_z$ of the control and ancilla is measured, with outcome m_1 . Next, the parity operator $\sigma_x \otimes \sigma_x$ of the ancilla and target is measured, with outcome m_2 . Finally, the ancilla is measured in the computational basis σ_z with outcome m_3 . The three outcomes determine the final corrective operation $\sigma_z^{m_2} \otimes \sigma_x^{m_1+m_3}$ on the control and target, which can also be done by updating the Pauli frame.

on a triangular lattice with up to six nearest neighbors for each qubit. Since the lower part of the hexagonal cell is a double T junction, it can be used for single-qubit Clifford gates by braiding. Note that if the physical implementation only allows for two orientations of the wires, as opposed to three, such hexagonal cells can also be embedded into a lattice with a brick-wall geometry, where hexagonal cells are equivalent to nine-island rectangular cells.

Braiding of Majoranas does not allow for a CNOT gate. However, qubit parity measurements and single-qubit Clifford gates can be used to construct a CNOT gate using an ancilla qubit [46]. Consider the quantum circuit shown in Fig. 4. The action of a CNOT gate is to flip the target qubit $|t\rangle$ if the control qubit $|c\rangle$ is in the $|1\rangle$ state and to apply the identity if it is in the $|0\rangle$ state. Using an ancilla qubit initialized in the state $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$, a CNOT gate can be implemented by a series of qubit parity measurements and some corrective operations. In the first step of the quantum circuit, the two-qubit parity operator $\sigma_z \otimes \sigma_z$

between the control and ancilla qubit is measured, yielding a measurement outcome $m_1 = 0$ for even and $m_1 = 1$ for odd parity. Next, the rotated parity operator $\sigma_x \otimes \sigma_x$ between the ancilla and target qubit is measured with outcome m_2 , which is equivalent to a $\sigma_z \otimes \sigma_z$ measurement with basis-rotating Hadamard gates applied before and after the measurement. Finally, the ancilla qubit is measured in the computational (σ_z) basis with outcome m_3 . The three measurement outcomes are used to determine the corrective operation on the control and target qubit $\sigma_z^{m_2} \otimes \sigma_x^{m_1+m_3}$. This procedure can be seen as a topological version of the nonlocal CNOT gates considered in Ref. [47].

Since the corrective operation consists of Pauli gates, and Pauli gates can be commuted past Clifford gates generating only other Pauli gates, it is not necessary to physically perform the actual gate corresponding to the correction. Therefore, as long as the gate circuit consists only of Clifford operations, Pauli gates only need to be tracked by a classical computer by updating the so-called Pauli frame, using a procedure known as Pauli tracking [48]. This is, strictly speaking, no longer the case when T gates are involved since $\sigma_x T = T^\dagger \sigma_x$. In this case, gate synthesis at later steps needs to replace T by T^\dagger when commuting σ_x past a T gate.

This parity-measurement-based protocol for a CNOT gate can be readily implemented in hexagonal-cell qubits (see Fig. 5). First, the ancilla qubit is initialized in the hexagonal cell occupied by the control qubit. After the application of Hadamard gates on the ancilla and target qubit (a), the two-qubit parity operator $\sigma_z \otimes \sigma_z$ of the control and ancilla qubit is measured by moving the first two Majoranas of each qubit onto three connected superconducting islands (b). Since the total fermion parity of the connected islands ($ic_1c_2)(ia_1a_2)$ is precisely the qubit parity operator, the

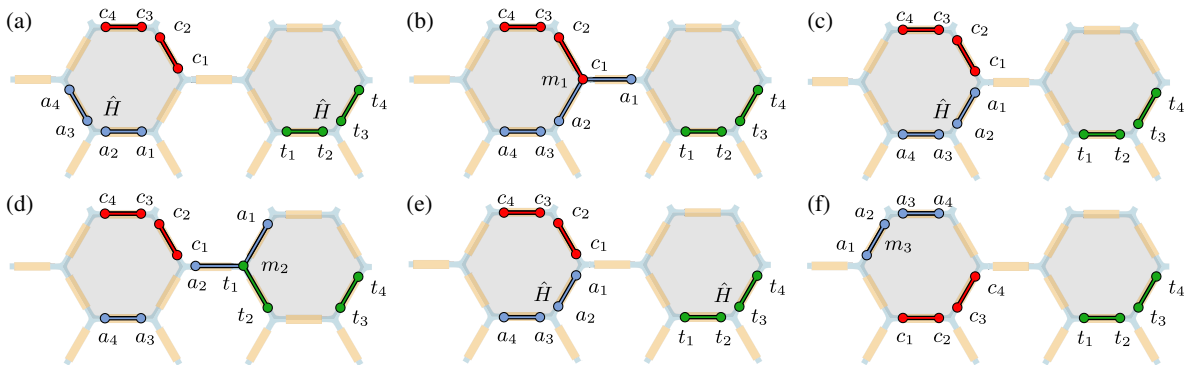


FIG. 5. Protocol for a CNOT between two adjacent hexagonal-cell qubits using the quantum circuit in Fig. 4. In the cell occupied by the control qubit (red), an ancilla (blue) is initialized in the $|0\rangle$ state and moved to the double T junction of the cell. (a) The ancilla and target (green) are rotated via a Hadamard gate. (b) The first two Majoranas of the control, c_1 and c_2 and ancilla a_1 and a_2 are moved onto a connected island, and the four-Majorana fermion parity $-a_1a_2c_1c_2$ is measured, corresponding to a two-qubit parity measurement $\sigma_z \otimes \sigma_z$ with outcome m_1 . (c) The ancilla is moved back to the double T junction for another H gate. (d) The ancilla and target parity m_2 is measured via a four-Majorana parity measurement in the right cell. (e) An H gate is applied to the ancilla and target qubits in their respective double T junctions. (f) Finally, all qubits return to their initial positions, and the ancilla qubit is measured by measuring the two-Majorana fermion parity ia_1a_2 with outcome m_3 .

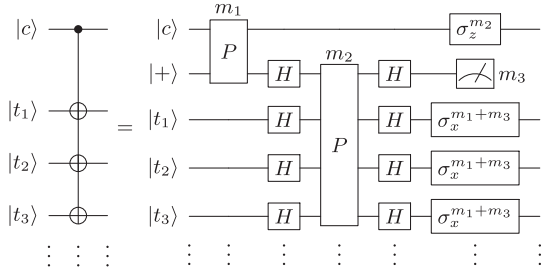


FIG. 6. Quantum circuit for a multitarget CNOT gate using parity measurements and an ancilla qubit initialized in the $|+\rangle$ state. First, the parity operator $\sigma_z \otimes \sigma_z$ of the control and ancilla is measured, with outcome m_1 . Next, the parity operator $\sigma_x \otimes \sigma_x^{\otimes n}$ of the ancilla and n targets is measured, with outcome m_2 . Finally, the ancilla is measured in the computational basis σ_z , with outcome m_3 . The three outcomes determine the final correctional operation $\sigma_z^{m_2} \otimes (\sigma_x^{m_1+m_3})^{\otimes n}$ on the control and targets, which can also be done by updating the Pauli frame.

measurement of the four-Majorana fermion parity yields the two-qubit parity. After another H gate (c), the same parity measurement is repeated for the ancilla and target qubit (d). After the final set of H gates, all Majoranas are returned to their initial positions, and the ancilla qubit is read out by measuring the two-Majorana parity ia_1a_2 . The ancilla qubit may be discarded after the protocol. This concludes the protocol for a CNOT gate between adjacent hexagonal-cell qubits. In Appendix A, we demonstrate that this scheme can also be used for CNOT gates between arbitrary hexagonal-cell qubits. Moreover, we show that multiple CNOT gates can be applied simultaneously in a transversal fashion.

This parity measurement-based protocol for a CNOT gate can be extended to multitarget CNOT gates. A multitarget CNOT gate corresponds to the application of n CNOT gates with one control qubit $|c\rangle$ and n different target qubits $|t_i\rangle$. Such multitarget CNOT gates are part of magic-state distillation protocols, which are used for the implementation of a robust logical T gate. Using the protocol in Fig. 4, a multitarget CNOT with n targets would require n ancilla qubits and $3n$ parity measurements. A faster alternative uses only one ancilla qubit and a parity measurement involving the ancilla and all n target qubits (see Fig. 6). This multitarget CNOT protocol replaces $3n$ measurements for n CNOTs by just three measurements for an n -qubit multitarget CNOT. A proof of the circuit identity in Fig. 6 is given in Appendix B. The application of this (transversal) multitarget CNOT gate to distillation protocols in topological superconductor networks is discussed in Sec. III.

B. T gates and stabilizer measurements

So far, we have only shown the implementation of the nonuniversal set of Clifford gates in topological superconductor networks. In fact, by virtue of the Gottesman-Knill theorem, Clifford quantum computers are no more

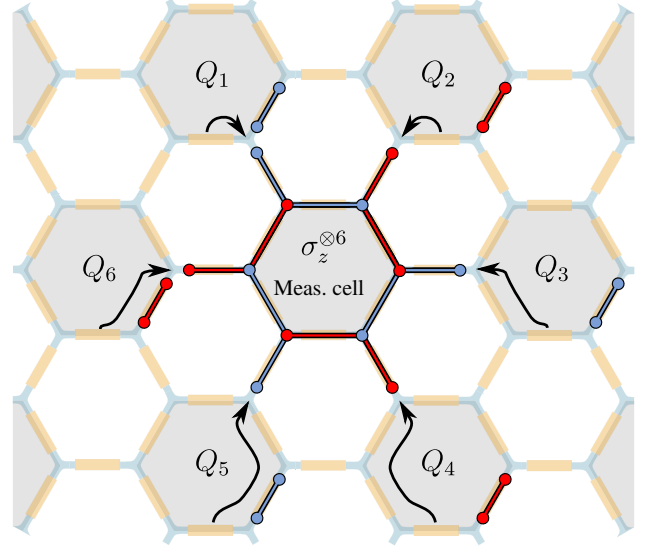


FIG. 7. Six-qubit parity measurement in a triangular lattice of hexagonal-cell qubits. Six hexagonal-cell qubits, Q_1 – Q_6 , are arranged around an empty cell that is used for the measurement of the parity operator $\sigma_z^{\otimes 6}$. For clarity, the Majoranas of each qubit are colored red and blue in an alternating fashion. The first two Majoranas of each qubit are moved to this cell, such that 12 connected superconductors host 12 Majoranas. The total 12-Majorana parity of this island is precisely the six-qubit parity operator.

powerful than classical computers [49]. Unfortunately, the T gate, which completes the universal gate set, cannot be done using a combination of braiding of Majoranas and parity measurements. An unprotected, error-prone T gate can be achieved by splitting the degeneracy of the parity states on the island hosting the first two Majoranas, such that the energy splitting between $|0\rangle$ and $|1\rangle$ is ΔE . After a time $\tau = \pi/4 \cdot \hbar/\Delta E$, the dynamic phase accumulated by time evolution will correspond to the T gate, and the degeneracy is restored again. In contrast to the Clifford gates, this protocol requires fine-tuning of the device control parameters and does not protect the T gate against errors. There exist more sophisticated protocols for physical T gates in Majorana-based setups [28,29], but for our purposes, any implementation of physical T gates is sufficient, as these gates can be used to implement T gates with arbitrary precision using the magic-state distillation procedure outlined in Sec. III.

In preparation for error correction using color codes, we also demonstrate the measurement of six-qubit parity operators $\sigma_z^{\otimes 6}$ without the need for ancilla qubits. Consider the six hexagonal-cell qubits in Fig. 7 arranged around an empty hexagonal cell. If the first two Majoranas of each surrounding qubit are moved onto 12 connected islands, the total parity of this island will be the 12-Majorana operator $\prod_{j=1}^6 i\gamma_{j,1}\gamma_{j,2}$, which is precisely the six-qubit parity operator $\sigma_z^{\otimes 6}$. This allows for the direct read-out

of the parity, circumventing the usual procedure [50] involving an ancilla qubit and six CNOTs between the ancilla and each qubit. The measurement of such n -qubit parity operators is required for quantum error correction, where they are the stabilizers of the code.

In summary, we have shown that topological superconductor networks, which allow for the movement of Majoranas, $2n$ -Majorana parity measurements, and tuning of the energy splitting between parity states, constitute universal quantum computers. In particular, triangular lattices of hexagonal-cell qubits feature topologically protected Clifford gates and a T gate requiring fine-tuning. Furthermore, n -qubit parity operators $\sigma_z^{\otimes n}$ can be measured without the need for ancilla qubits, and multitarget CNOT gates require only three parity measurements, regardless of the number of target qubits.

III. TOPOLOGICAL SOFTWARE: DIAMOND COLOR CODES

Unless the topological hardware is perfect, qubit errors will occur after a certain number of gate operations. These errors change the outcome of the quantum computation and therefore need to be actively corrected. In quantum error correction, multiple physical qubits are used to encode a single error-resilient logical qubit. In so-called stabilizer codes [3,51], the logical qubit is encoded in the degenerate ground-state space of a Hamiltonian,

$$H_S = -\sum_i \mathcal{O}_i, \quad [\mathcal{O}_i, \mathcal{O}_j] = 0. \quad (6)$$

Here, \mathcal{O}_i are operators with eigenvalues ± 1 , which are called stabilizers and are products of Pauli operators. Since all stabilizers commute, the ground-state space is spanned by the simultaneous $+1$ eigenstates of all stabilizers, under the condition that the operator -1 is not part of the stabilizer group. Logical information can be stored in this degenerate ground-state space, also referred to as code space. For the logical qubits discussed in this work, the ground-state space is doubly degenerate, where the eigenstates define the logical qubit states $|0_L\rangle$ and $|1_L\rangle$. Note that the Hamiltonian H_S does not necessarily describe the physical system used for quantum computation. Instead, H_S merely defines the code space, into which the physical system is projected by measuring all stabilizer operators.

Errors occurring on physical qubits will change the eigenvalue of certain stabilizers. The so-called code distance is the minimum number of qubits that need to be affected by errors in order to change the logical subspace, i.e., map $|0_L\rangle$ onto $|1_L\rangle$ and vice versa. In order to prevent this from happening, all stabilizer operators are measured periodically before physical errors can affect the encoded information. These measurements reveal the so-called error syndrome, which is a list of all stabilizer measurement outcomes ± 1 . This information is used to correct the errors

that have occurred. The practical problem that has to be overcome is that only the syndrome is available, while the actual errors are unknown. Moreover, different error configurations can lead to the same error syndrome. The classical algorithm that finds a suitable error configuration belonging to a given syndrome is called a decoder [25,52–58].

Typically, quantum error-correcting codes operate in code cycles. In every code cycle, logical operations are performed, the syndrome is read out by making use of stabilizer measurements, and the errors on physical qubits are actively corrected. But even logical qubits only have a finite survival time, as quantum error-correcting codes merely replace a physical error rate by a (preferably lower) logical error rate. The minimum number of physical qubits that need to be affected by errors within a code cycle, such that the errors are no longer correctable, scales with the code distance. There are two prescriptions for how a higher-distance code can be obtained from a low-distance code: code concatenation [51] and topological codes. Code concatenation has the drawback that it requires the measurement of increasingly nonlocal stabilizer operators with increasing code distance. In contrast, the stabilizers of topological codes remain spatially local as the code distance is increased. Moreover, in topological codes, the encoded logical quantum information is protected from local perturbations because virtual transitions require an order in perturbation proportional to the system size. In the case of surface and color codes, errors generate and propagate anyons—excitations of the system with nontrivial braiding statistics—that are manifested in a changed stabilizer measurement outcome. This implies that for surface and color codes defined on a lattice, anyons need to propagate through the entire lattice in order to affect the logical subspace, i.e., errors need to form along a nontrivial line through the lattice. The locality of stabilizers and high error resilience are the two key advantages that distinguish topological from nontopological codes.

In fault-tolerant quantum computing, it is desirable to perform all gate operations on the level of encoded logical qubits without the need to decode them back to error-prone physical qubits [16]. However, the physical operations that constitute a logical gate U_L are typically entirely different from the known physical gates U . An exception are so-called transversal gates, which, for our purposes, are logical gates that are precisely the application of the corresponding physical gate (or its Hermitian conjugate) on each qubit, i.e., $U_L = U^{(\dagger)\otimes n}$. This has the advantage that errors due to faulty implementations of single physical gates do not spread to other physical qubits. Moreover, transversal gates directly employ physical gates to implement logical gates, enabling us to carry over the topological protection of physical gates to the level of logical gates. However, the Eastin-Knill theorem states that no code can have a set of transversal gates that is also a universal gate set [22].

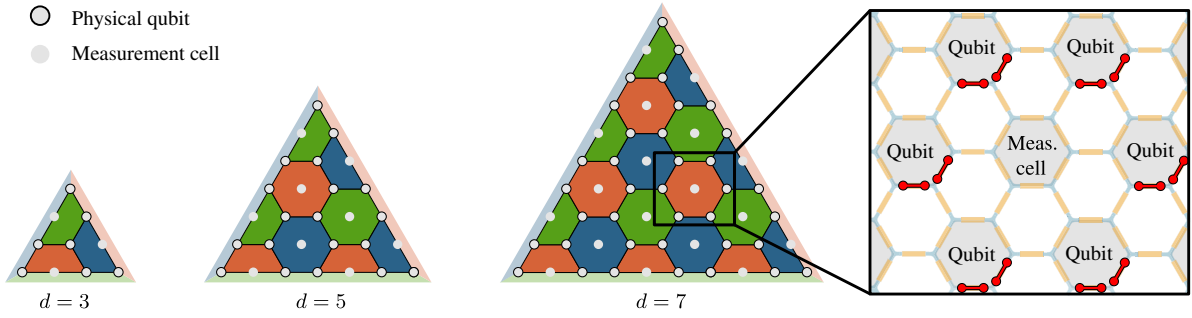


FIG. 8. First three topological triangular color codes with code distances 3, 5, and 7 (where the smallest one is equivalent to the Steane code [63]). These 6.6.6 color codes are defined on a hexagonal lattice, where each vertex is a physical qubit and each face is an X -type and a Z -type stabilizer involving the surrounding qubits. Physical errors on a qubit affect the three different-colored stabilizers and edges surrounding the qubit. In the triangular lattice of hexagonal-cell qubits, the empty cell in the center of each face can be used for stabilizer measurement.

One family of topological codes with transversal gates are topological color codes [26]. Their set of transversal gates are the Clifford gates. Since this set coincides with the set of topologically protected operations of topological superconductor networks, color codes are a natural fit to Majorana-based hardware. In comparison to the closely related [59] surface codes, which cannot implement braiding transversally, color codes also feature a higher error threshold. The error threshold is the maximum physical error rate below which logical errors are suppressed by increasing the code size, allowing for quantum computation of arbitrary duration. We note that in circuit models where Clifford gates are error prone and stabilizers are measured using ancilla qubits and CNOT gates, surface codes indeed feature a higher threshold than color codes [60]. However, in the limit of topological hardware where Clifford operations have a vanishing error rate and stabilizer read-out does not require ancilla qubits, color codes outperform surface codes even in the presence of measurement errors during syndrome read-out [61,62]. In addition, since the Clifford gates are transversal for color codes, their implementation only requires one code cycle. This reduces time overhead compared to surface codes, where their implementation requires multiple code cycles [50].

A. Triangular and diamond color codes

Color codes are stabilizer codes that are defined on lattices with three colorable faces. Physical qubits sit on the vertices, and the stabilizers are operators acting on all qubits surrounding a face. Figure 8 shows a family of color codes that is defined on a hexagonal lattice of physical qubits, namely, the triangular 6.6.6 color codes. Here, all stabilizers involve either four or six qubits. There are two stabilizers per face f , an X -type stabilizer $\mathcal{O}_X = \otimes_{i \in f} \sigma_x$ and a Z -type stabilizer $\mathcal{O}_Z = \otimes_{i \in f} \sigma_z$. Thus, the logical qubits in the color code are encoded in the ground-state space of the Hamiltonian,

$$H_{\text{color code}} = -\sum_{\text{faces}} \mathcal{O}_X - \sum_{\text{faces}} \mathcal{O}_Z. \quad (7)$$

To initialize a color-code qubit in the logical $|0_L\rangle$ state, all physical qubits are initialized in the $|0\rangle$ state, the stabilizers are measured, and the errors are corrected.

Every physical qubit is part of up to three different-colored X -type and Z -type stabilizers. At the boundaries, qubits are only part of one or two stabilizers, but if one assigns colors to the boundaries (see Fig. 8), every qubit is part of three different-colored stabilizers *or* boundaries. A σ_z -type Pauli error on a physical qubit will flip the three surrounding red, green, and blue X -type stabilizers. Conversely, a σ_x -type error will flip three Z -type stabilizers. In the language of topological codes, flipped stabilizers with eigenvalue -1 host an anyon. Thus, errors generate and propagate strings with red, green, and blue anyons at their endpoints. Each edge can absorb anyons of its respective color. A logical error occurs when physical errors propagate a red, a green, and a blue anyon to the red, green, and blue edges, respectively. Thus, a logical $(\sigma_z)_L$ operator is given by any string of physical σ_z operators that propagates anyons in this way. In particular, physical σ_z operators on all physical qubits sitting on any one of the three edges propagate anyons accordingly and therefore correspond to logical $(\sigma_z)_L$ operators. Similarly, logical $(\sigma_x)_L$ operators correspond to strings of physical σ_x operators.

Each code cycle consists of three steps. First, logical operations are performed on the encoded qubits. Next, the error syndrome is extracted by measuring all stabilizers. The syndrome is then given to the decoder. Finally, the corrections proposed by the decoder are applied. Note that it is not necessary to physically correct the errors, as they can be handled classically by Pauli tracking [48], under the assumptions discussed in Sec. II.

In the triangular lattice formed by hexagonal-cell qubits, the cell in the center of each stabilizer is not occupied by a

physical qubit. Instead, these cells can be used for stabilizer measurements, as shown in Fig. 7 for Z-type stabilizers. Note that X-type stabilizers can be measured by applying a Hadamard gate to all qubits before and after the measurement. Color codes fall into the class of CSS codes [63,64]; i.e., all stabilizers are products of only σ_z operators or only σ_x operators. CSS codes have a transversal implementation of the CNOT gate, where the logical CNOT gate corresponds to the application of physical CNOTs between all corresponding physical qubits of two codes [see Fig. 9(a)]. Moreover, color codes are *strong* CSS codes because the support of Z-type and X-type stabilizers coincides. This implies that Hadamard gates are transversal, and the logical Hadamard gate $H_L = H^{\otimes n}$ maps stabilizer states onto other stabilizer states. In general, this is not true for the application of physical S gates on all qubits. Therefore, the transversal S_L gate requires greater care, as some physical S gates need to be replaced by S^\dagger gates. One general prescription is to bicolor the vertices of the color-code graph, such that neighboring qubits have different colors. In Fig. 9(a), we color the sublattice containing the corner qubits blue, and we color the other sublattice orange. The logical S_L gate then corresponds to physical S gates on blue qubits and physical S^\dagger gates on orange qubits [65].

All physical operations required for single-qubit transversal gates can be applied simultaneously since they only require braiding within each hexagonal-cell qubit. As we show in Appendix A, also for transversal CNOTs, all physical CNOTs can be performed simultaneously in a hexagonal-cell qubit geometry. However, this requires the triangles encoding the control and target qubit to be oriented the same way. Thus, the densest packing of triangular color codes along one line is not practical. Instead, we choose to extend the upward-pointing

triangular codes into the unused space on their right, forming diamonds, as shown in Fig. 9(b). Since all stabilizers of one type and color can be measured simultaneously, this happens at no increase in space or time overhead. Moreover, our Monte Carlo simulation in Sec. V shows that diamond codes even feature a lower logical error rate compared to triangular codes with the same code distance. Note that when extending triangles to diamonds, only one of the edges becomes longer compared to the triangular code. As the code distance is given by the length of the *shortest* edge, the extension to diamond color codes lowers the logical error rate despite leaving the code distance unchanged.

Universal fault-tolerant quantum computation with logical diamond color-code qubits requires the implementation of a universal gate set $\{H_L, S_L, \text{CNOT}_L, T_L\}$. The first two gates are implemented directly in a transversal fashion. The CNOT_L gate requires special care. Even though it can be done transversally, CNOTs in hexagonal-cell qubits use physical ancilla qubits, which are not protected against noise. In the remainder of this section, we show that a one-dimensional arrangement of data qubits with a magic-state distillery above and a CNOT bypass below [see Fig. 9(c)] implements the remaining two logical gates in a fault-tolerant fashion. A magic-state distillery is an array of qubits used for magic-state distillation, whereas data qubits are qubits used for quantum computation but not for distillation. We present protocols that use the CNOT bypass to implement a fault-tolerant CNOT_L gate with an overhead that scales with neither the code distance nor the distance between the control and target qubits. Furthermore, we demonstrate how the magic-state distillery can be used to produce and store magic states, which allow for a fault-tolerant implementation of the T_L gate.

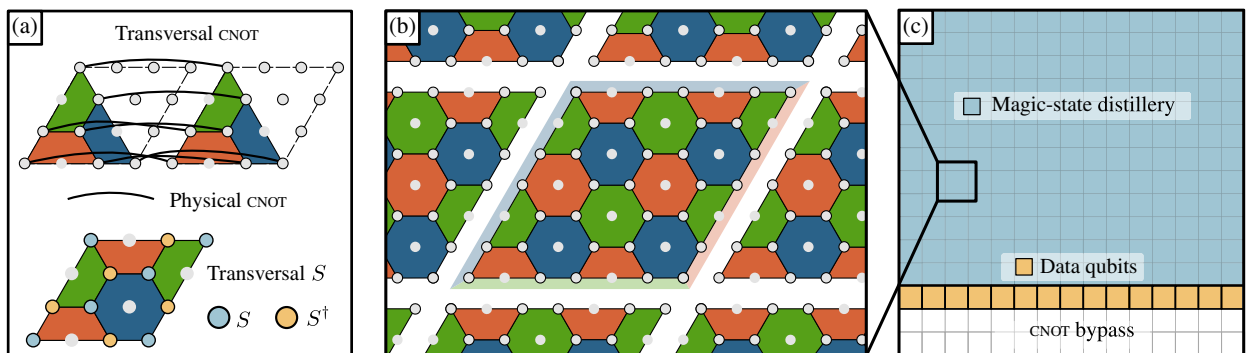


FIG. 9. (a) Color codes feature transversal Clifford gates. While the logical Hadamard gate is simply $H_L = H^{\otimes n}$, the logical S gate S_L is a mixture of physical S and S^\dagger gates. Using a bicoloration of the physical qubits, such that the sublattice involving the corner qubits is blue and the other one is orange, S_L requires physical S gates on blue qubits and an S^\dagger gate on orange qubits. The logical CNOT gate corresponds to n physical CNOTs between pairs of qubits from two triangles. (b) Since this requires the movement of ancilla qubits from one triangle to the other, this leaves some unused space in between, which can be used for the diamond color codes. These form a square lattice of logical qubits. (c) Universal fault-tolerant quantum computation can be achieved on a line of data qubits with a magic-state distillery and a CNOT bypass.

B. Logical CNOT gates

We present three protocols for logical CNOT gates between data qubits. In the first protocol, the control qubit is moved to the target qubit through the CNOT bypass, and the CNOT gate is performed transversally [see Fig. 9(a)]. However, the ancilla qubits in this protocol are physical qubits and therefore susceptible to errors. Moreover, since measurements are part of the CNOT protocol, a physical CNOT gate may introduce additional errors if measurements are not perfect. Both factors increase the noise level for logical CNOT gates. The noise level can be decreased by substituting physical ancilla qubits by a logical ancilla qubit.

An implementation of the circuit in Fig. 4 with logical qubits requires parity measurements between logical qubits. Since the logical σ_z operator is a nontrivial string of physical σ_z operators through the code, the two-qubit ZZ-parity operator of two distance d codes is a product of at least $2d$ σ_z operators. One method of fault tolerantly and projectively measuring the two-qubit parities of logical qubits is called lattice surgery [34]. Here, new stabilizers are temporarily introduced on the boundary between two logical qubits. In Fig. 10(a), we show a lattice surgery protocol along the green boundaries of two diamond

color-code qubits, although any two boundaries can be used regardless of color as long as they have equal lengths. In this protocol, the red four-term X stabilizers at the boundaries are merged to form eight-term stabilizers. The corresponding Z stabilizers remain unchanged (see Fig. 21 in Appendix C). Green three- and four-term stabilizers are introduced which commute with all other stabilizers and involve each boundary qubit exactly once. Therefore, these stabilizers are only measured in the Z basis, as the product of all green boundary stabilizers is precisely the ZZ parity. If these stabilizers are measured along with the other stabilizers, qubit errors can be corrected and the parity measurement is fault tolerant. The error due to faulty measurements can be reduced by repeating sufficiently many rounds of syndrome extraction. After the product of the green boundary stabilizers is determined—and therefore the two-qubit parity—the stabilizers are reverted to the initial configuration. Similarly, the XX parity can be obtained by swapping X and Z stabilizers in the aforementioned protocol. We stress that the lattice surgery protocol projectively measures the logical two-qubit parity without revealing any additional information, as we discuss in greater detail in Appendix C.

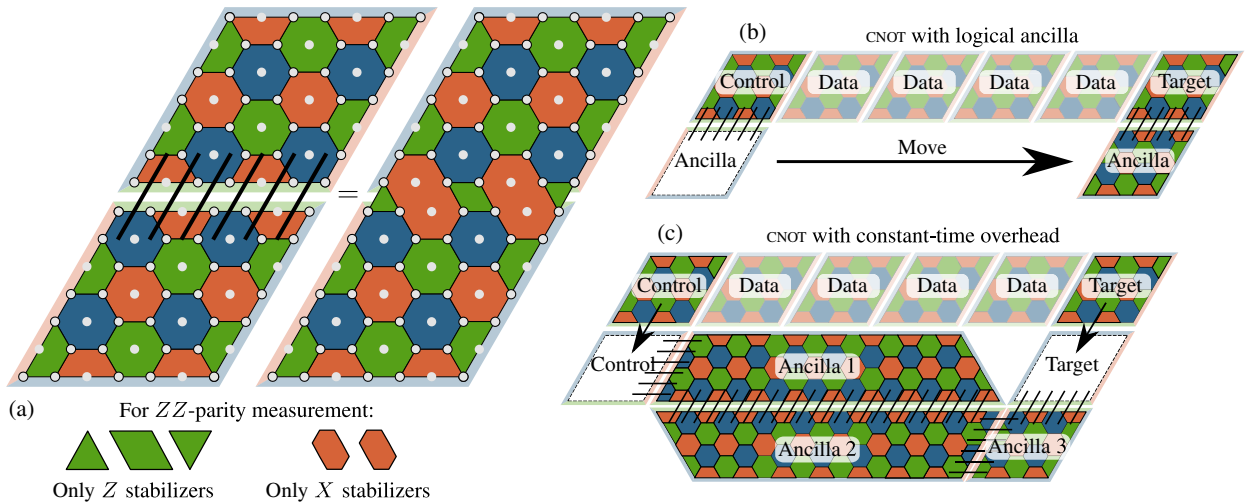


FIG. 10. (a) Fault-tolerant ZZ-parity (XX -parity) measurement between two diamond color-code qubits by lattice surgery [34], denoted by black lines crossing the neighboring boundaries. First, new three- and four-qubit green stabilizers are introduced, and new eight-qubit stabilizers are obtained by merging red plaquettes along the boundary. These stabilizers are measured along with all other stabilizers in order to obtain the ZZ parity (XX parity), where the new green boundary stabilizers are *only* measured in the Z basis (X basis) and the red eight-qubit stabilizers only in the X basis (Z basis). (All stabilizers are explicitly shown in Fig. 21 in Appendix C.) The product of the green boundary stabilizers is precisely the two-qubit parity. Finally, the stabilizers are returned to their initial configuration before the lattice surgery. (b) Protocol for a fault-tolerant CNOT using lattice surgery. A logical ancilla is initialized in the $|+\rangle$ state. The ZZ parity between the control and ancilla is measured, and the ancilla is moved through the CNOT bypass to the target. Finally, the XX parity between the ancilla and target is measured, and the ancilla is read out. The length of this protocol scales linearly with the distance between the control and target. (c) Lattice-surgery-based CNOT protocol with constant-time overhead. The ZZ parities between the control and three ancilla qubits in the $|+\rangle$ state are measured simultaneously using the three lattice surgeries indicated in the figure. Next, the XX parity between ancilla 3 and the target is measured. Finally, ancilla 3 is read out in the Z basis, while ancillas 1 and 2 are measured in the X basis. In the presence of measurement errors during syndrome read-out, this protocol scales logarithmically with the distance between the control and target.

Thus, the second protocol is a CNOT with a logical ancilla shown in Fig. 10(b). A logical $|+\rangle$ ancilla is initialized in the CNOT bypass next to the control qubit. The ZZ parity between the ancilla and control is measured by lattice surgery, and the ancilla is moved to the target qubit. Finally, the XX parity between the ancilla and target is measured, and the ancilla is measured in the σ_z basis. Although this protocol yields a logical CNOT gate with arbitrary precision, it still has one major drawback: The protocol length increases linearly with the distance between the control and target qubit.

This can be alleviated by using two additional ancilla qubits with long edges, which replaces the movement of the ancilla qubit by a number of simultaneous stabilizer measurements at the long edge. The third protocol is a CNOT with constant-time overhead [see Fig. 10(c)]. Three $|+\rangle$ ancillas are arranged such that ancillas 1 and 2 both have a short and a long edge and cover the entire distance between the control and target qubit. Using lattice surgery, the ZZ parities between the control and ancilla 1, between ancillas 1 and 2, and between ancillas 2 and 3 can be measured simultaneously. This is equivalent to measuring the two-qubit parities between the control qubit and each of the ancilla qubits. Therefore, ancilla 3 can be directly used as the CNOT ancilla. Its XX parity with the target qubit is measured, and it is read out in the σ_z basis. Ancillas 1 and 2 cannot be discarded right away, as they are still entangled with the control qubit. They can be disentangled by measuring the ancillas in the σ_x basis with measurement outcomes m_1 and m_2 , and by applying a $\sigma_z^{m_1+m_2}$ correction to the control qubit. An explanation of the quantum circuit corresponding to this protocol is found in Appendix E.

In the absence of measurement errors, this protocol has a constant-time overhead. This is no longer true if syndrome measurements are faulty. Such a measurement can be described by a perfect measurement, followed by the identity map with probability p and a flipped outcome with probability $1 - p$. Since more boundary stabilizers are involved in the parity measurement comprising ancillas 1 and 2, they need to be measured more often to achieve the same accuracy as the other parity measurements. However, because the measurement error probability decreases exponentially with each repetition, whereas the number of boundary stabilizers only increases linearly with the distance between the control and target, the time overhead of this CNOT only scales logarithmically with the control-target distance.

We have presented three protocols for logical CNOT gates. The transversal protocol between nearest neighbors is fast but has a fixed accuracy and a time overhead that scales linearly with the control-target separation. The second protocol uses a logical ancilla and can therefore achieve arbitrary accuracy, but it is slower than the first protocol as it requires multiple code cycles. The third protocol eliminates the time overhead or replaces it by a

time overhead that scales favorably as the logarithm of the distance between the control and target. By adding rows to the CNOT bypass, multiple spatially intertwined CNOT gates can be performed simultaneously. Note that due to the overhead in quantum wires in a hexagonal-cell qubit, logical diamond color-code qubits do not block each other's paths when moving, as they can be moved through one another, similar to how ancilla qubits can be moved past other qubits in the transversal CNOT protocols of Figs. 5 and 20.

C. Magic-state distillation

The only gate remaining for a universal fault-tolerant quantum computer is the logical T_L gate. We point out that even if the physical hardware had a topologically protected physical T gate, there would be no way of directly using this for a T_L gate as the T gate cannot be transversal in a code with transversal Clifford gates due to the Eastin-Knill theorem [22], which states that the ability of a quantum code to detect arbitrary errors on any single physical subsystem is incompatible with the existence of a universal, transversal encoded gate set for the code. There exist code-switching methods that allow us to switch the logical qubit from one code to another code with a different set of nonuniversal transversal gates. However, in order for this set to include the T gate and for the stabilizers to still remain local, the qubits need to be arranged in *three* dimensions instead of two [66].

One possibility to implement a logical low-error T gate using logical Clifford gates and a physical T gate is magic-state distillation. Consider the state injection circuit shown in Fig. 11, which is equivalent to a T gate on the qubit $|\psi\rangle$. Using an ancilla magic state $|m\rangle = T|+\rangle = (|0\rangle + e^{i\pi/4}|1\rangle)/\sqrt{2}$, a CNOT between the qubit and one prepared in a magic state, followed by the measurement of the magic state with outcome m_z , corresponds to a T gate up to a correctional S^{m_z} operation. Such a procedure of effectively generating a quantum gate by making use of suitable quantum state resources is referred to as gate teleportation.

In order for this state injection algorithm to yield a logical T gate, the magic state $|m\rangle$ needs to be an encoded logical qubit. However, since the physical T gate is not topologically protected and physical qubits are not protected against errors during the encoding process, we can only generate faulty magic states that are well

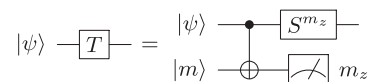


FIG. 11. State injection algorithm: A CNOT between a qubit $|\psi\rangle$ and a magic state $|m\rangle = (|0\rangle + e^{i\pi/4}|1\rangle)/\sqrt{2}$, followed by a measurement of $|m\rangle$ with outcome m_z and a correctional S^{m_z} gate, is equivalent to a T gate on the qubit.

approximated by $|\tilde{m}\rangle = (|0\rangle + e^{i(\pi/4+\epsilon)}|1\rangle)/\sqrt{2}$, even though further errors are expected and allowed for. Magic-state distillation is an algorithm deeply related to quantum error correction that generates low-error magic states using many faulty magic states $|\tilde{m}\rangle$ with angle deviations ϵ of up to 17.3% [30]. Many such algorithms exist, such as a 15-to-1 protocol [30], a 10-to-2 protocol [67], or, more generally for an integer k , a $3k + 8$ -to- k protocol [68,69]. These protocols only require (transversal) Clifford gates, in particular, multitarget CNOT gates. Combinations of these protocols [68] can be used to generate magic states—and therefore effectively T gates—with the desired precision.

Logical magic states can be encoded from physical magic states using a variant of the code injection procedure described in Ref. [34]. In Fig. 12, we depict this procedure for a diamond color code. A detailed explanation of the presented protocol is given in Appendix D. The protocol can correct errors on any pair of physical qubits, but certain errors with support on three qubits cause the injection of a faulty state, regardless of the code distance of the diamond code used. This further substantiates the need for magic-state distillation.

In principle, the multitarget CNOTs in the distillation protocols can be done using many iterations of the logical CNOT gates that we discussed previously. However, for logical CNOTs between data qubits, we focused on the operations having a low error rate. Since distillation protocols are only performed once, and afterwards magic-state qubits are merely stored until their use, the priority of their multitarget CNOTs should be speed over accuracy of individual gates, such that magic states can be distilled fast.

Majorana-based qubits offer the possibility of a fast multitarget CNOT gate using the protocol in Fig. 6. Even though this gate is transversal for color-code qubits, the parity measurements involve physical qubits that are spatially separated—i.e., every first physical qubit of each involved logical qubit, every second physical qubit, and so on. One method to bring them closer together is by

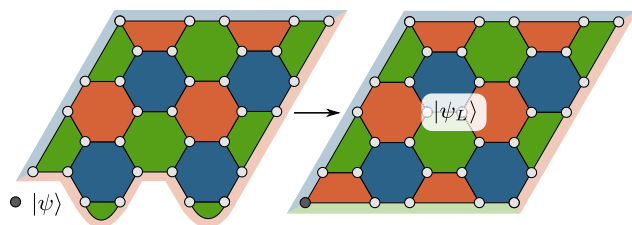


FIG. 12. Code injection procedure which encodes an unknown physical state $|\psi\rangle$ (gray qubit) into a logical state $|\psi_L\rangle$. First, the stabilizer state in the left panel is prepared by measuring all the stabilizers shown. Finally, we cease measuring the green stabilizers at the bottom boundary and start measuring the red stabilizers.

rearranging the physical qubits using the inflation protocol shown in Fig. 13 for the example of four logical qubits arranged on a 2×2 grid. The protocol effectively rearranges the physical qubits of four logical qubits, such that they form blocks of four physical qubits that are part of multitarget CNOT gates. The analogous protocol with 15 qubits arranged on a 4×4 grid can be used for the transversal multitarget CNOTs required for 15-to-1 distillation. After sufficiently many rounds of magic-state distillation, the magic state is ready for state injection via a CNOT gate using any of the protocols outlined in the previous subsection.

Clifford gates and magic-state distillation operate independently from each other. In other words, during the application of Clifford gates on the data qubits in the quantum computation, magic states can be distilled in parallel and stored for later use in the magic-state distillery. Magic states can even be prepared offline and stored for future quantum computations. Since magic-state distillation is the part of the quantum computation that requires the greatest effort, magic states are resource states for quantum computation. With predestilled magic states, any quantum computation reduces to the application of (constant-time overhead) logical Clifford gates.

In conclusion, we have constructed logical diamond-shaped color-code qubits with transversal Clifford gates. Arranged on a line with a CNOT bypass and a magic-state distillery, they feature a robust T gate and a CNOT gate with constant-time overhead. The single-qubit Clifford gates are topologically protected because of the protection of the topological superconductor network. We note that apart from transversal CNOTs and fast multitarget CNOTs, the remaining protocols make no use of the diamond shape. In fact, if for data qubits one abandons the fast transversal CNOT protocol, each diamond-shaped data qubit can be replaced by two triangular color-code qubits with a straightforward generalization of the lattice surgery protocols. This reduces the spatial overhead for data qubits by a factor of 2, but it also slightly increases the logical error

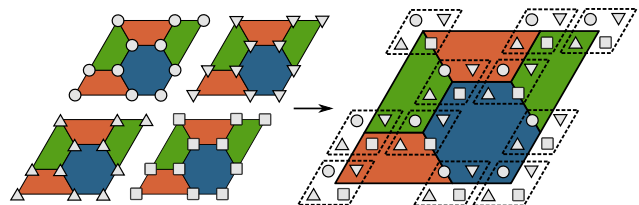


FIG. 13. Inflation protocol for transversal multitarget CNOT gates with four logical qubits. This protocol rearranges the physical qubits such that the qubits involved in transversal multitarget CNOT gates are now close to each other, i.e., every first physical qubit of each of the four logical qubits, every second, etc. This protocol can be used for the multitarget CNOTs required for magic-state distillation, e.g., using inflation of 15 qubits arranged on a 4×4 grid for 15-to-1 distillation.

rate. The same is not true for magic-state distillery qubits, as the inflation protocol for fast distillation still benefits from diamond color codes.

IV. PHYSICAL ARCHITECTURE: MAJORANA COOPER PAIR BOXES

In the previous sections, we have demonstrated that we can construct a fault-tolerant universal topological quantum computer on the basis of a topological superconductor network. Our construction requires that Majoranas can be moved, their parities measured, and the degeneracy of their parity states lifted. In this section, we review how this can be achieved using Majorana Cooper pair boxes following the scheme suggested in Ref. [35] (see also Refs. [18, 38–40]). While here we follow Ref. [35], other implementations of Majorana qubits can also be combined with a color code as discussed in this paper, as long as these architectures are capable of the three required operations. For instance, this scheme can, in principle, also be realized in Majorana box qubits [70] and related setups [71], where Majoranas are not moved directly by coupling neighboring islands but via braiding by measurement [72,73].

Pairs of Majorana zero modes can emerge at the ends of one-dimensional spinless p -wave superconductors [10]. Even though there are candidates for p -wave superconductors such as Sr_2RuO_4 [74], ordinary superconductors exhibit s -wave pairing. To effectively obtain the required p -wave pairing from s -wave pairing—and thereby Majorana zero modes—three essential ingredients are required (see Fig. 14): an s -wave superconductor, spin-orbit coupling, and one-dimensional spin-polarized conducting channels [4–9]. Experiments have focused on realizing this by using nanowires [11–14] or by appropriate patterning of two-dimensional electron gases [15,75], but in principle, this could also be achieved in edge states of quantum Hall, quantum spin Hall, or quantum anomalous Hall systems [4,76–79].

Unlike in ordinary s -wave superconductors, where the minimal excitation energy is given by the pairing gap Δ , the Majoranas have zero excitation energy. Each pair of Majoranas combines into a complex fermion that can be empty or occupied. Unpaired electrons can occupy these fermionic states at zero energy cost. When the island has one Majorana at each end, there is one complex zero-energy fermion. The occupation of this energy level is associated with the fermion parity of the mesoscopic island; i.e., the level is unoccupied for even and occupied for odd fermion parity. These statements hold true when the Majorana wire is proximity coupled to a grounded s -wave superconductor. If the superconductor is floating, the combined system of wire and proximity-coupled superconductor has a finite charging energy, which will, in general, lift the degeneracy between the even and odd-parity states [80–82].

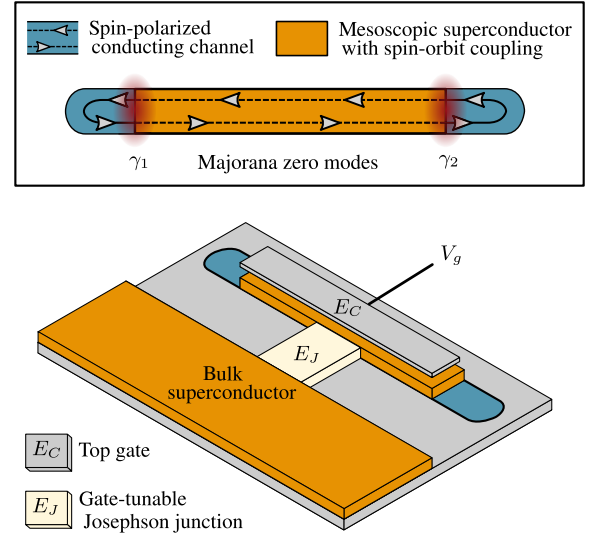


FIG. 14. A Majorana Cooper pair box as a basic building block of the topological hardware. Top diagram: A pair of Majorana zero modes γ_1 and γ_2 at the ends of a p -wave superconductor can be effectively obtained by depositing an s -wave superconductor with strong spin-orbit coupling on top of a material with a single spin-polarized conducting channel, such as a semiconducting nanowire in a magnetic field, a quantum anomalous Hall insulator, or a 2DEG in a strong magnetic field. Bottom diagram: A Majorana Cooper box requires the addition of charging energy E_C and Josephson energy E_J on the mesoscopic superconducting island. A top gate that is capacitively coupled to the superconducting island imposes a certain total charge on the island governed by the gate voltage V_g and the (fixed) charging energy. Furthermore, a bulk superconductor is Josephson coupled to the mesoscopic island through a gate-tunable Josephson junction, which tunes the Josephson energy and imposes a certain phase on the island.

A powerful scheme to manipulate Majorana zero modes exploits Majorana Cooper pair boxes (see Fig. 14) [35]. A gated wire coated by a superconducting island is coupled to a bulk superconductor through a tunable Josephson junction. Opening the Josephson junction effectively grounds the island, which will then support a Majorana degeneracy. This degeneracy will be progressively lifted by Coulomb charging effects as the Josephson coupling is reduced.

The low-energy Hamiltonian of the Majorana Cooper pair box [35] is given by the sum $H = H_C + H_J$ of a charging term

$$H_C = E_C (\hat{N} - N_0)^2 \quad (8)$$

with charging energy E_C , and a Josephson term

$$H_J = -E_J \cos \hat{\phi}, \quad (9)$$

with Josephson energy E_J . Here, \hat{N} is the operator that counts the electrons on the island and $N_0 = eV_g/(2E_C)$ is

the background charge controlled by the gate voltage V_g applied to the capacitively coupled gate. The operator $\hat{\varphi}$ is the phase of the superconducting island, obeying the commutation relation $[\hat{\varphi}, \hat{N}] = 2i$. Even and odd-parity states obey periodic and antiperiodic boundary conditions when writing the wave function in the phase representation [80].

Figure 15 shows the spectrum of H in three characteristic regimes [35]. In the Majorana regime $E_C \ll E_J$, the phase $\hat{\varphi}$ is fixed by the bulk superconductor, and the spectrum is almost V_g independent. In this regime, there are two nearly degenerate ground states whose splitting ΔE is exponentially small in E_J/E_C . These ground states are separated from excited states by an energy $\sim \sqrt{E_J E_C}$. In the opposite Coulomb regime $E_C \gg E_J$, the eigenstates are well-defined charge states. The two lowest charge states with even and odd parity are split for all values of V_g , except at the charge degeneracy points where N_0 is half integer. Depending on whether N_0 is closer to an even or odd integer, the ground state has either even or odd fermion parity. Thus, one can impose a desired fermion parity on the state of the Majorana Cooper pair box by tuning it to the Coulomb regime and relaxing to the ground state. The intermediate regime with $E_C \sim E_J$ can be understood starting from the Majorana regime as the result of Coulomb charging lifting the ground-state degeneracy or from the Coulomb regime as a result of forming avoided crossings between states of equal fermion parity by Cooper pair tunneling in and out of the island.

Using these three regimes of the Majorana Cooper pair box, all operations required for color-code quantum computing with a topological superconductor network can be implemented. In a network, islands hosting Majoranas that encode a qubit are tuned to the Majorana regime, such that the parity states—and therefore the encoded qubits—are degenerate. All other (empty) islands are tuned to the Coulomb regime. The remainder of this section is devoted to showing how to use these two regimes to move Majoranas through the network and how to employ the intermediate regime for parity measurements [35]. This is complemented by degeneracy splitting, which is straightforwardly implemented by decreasing E_J on an island.

A. Moving Majoranas

Neighboring Majorana Cooper pair boxes with individually controllable gate voltage and Josephson energy are connected via tunnel coupling (see Fig. 16). The interisland transmission probability τ can be controlled by a pincher gate located between the islands. Following Ref. [35], this junction can be used to move Majoranas between islands. Starting with two decoupled islands ($\tau = 0$) in the Majorana (left island) and Coulomb (right island) regime (see Fig. 17), γ_2 is moved to the right island by increasing the interisland coupling and then tuning the right island to the Majorana regime by increasing its Josephson coupling to the bulk superconductor. This places the system into an eigenstate of the *total* parity $i\gamma_1\gamma_2$ of both islands. One should ensure that at the beginning of the protocol, the right island is initialized into the even-parity sector by tuning V_g

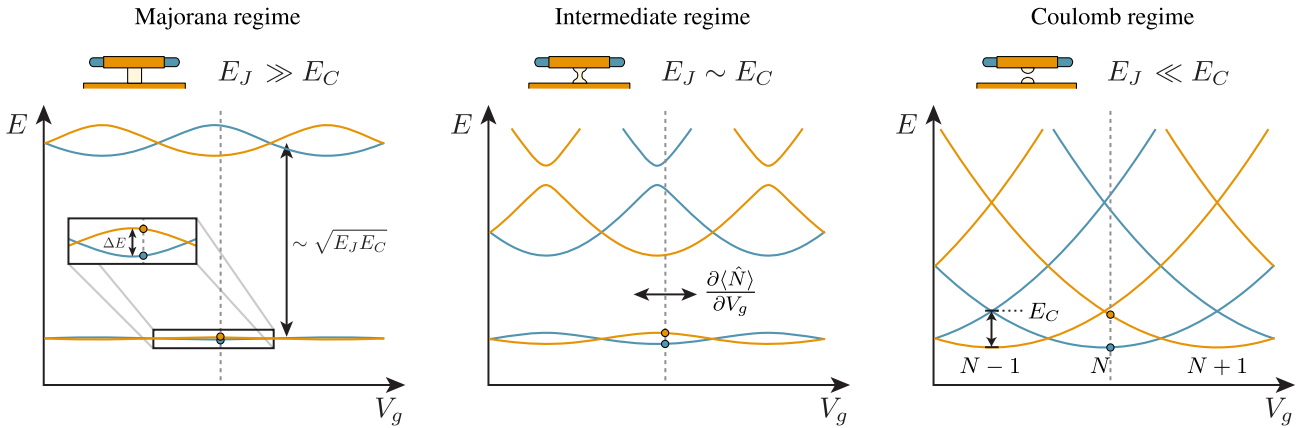


FIG. 15. Parity-to-charge conversion in the Majorana Cooper pair box, as described in Ref. [35], and energy levels of the Majorana Cooper pair box $H_C + H_J$ as a function of gate voltage V_g and for different ratios E_J/E_C . In the Majorana regime $E_J \gg E_C$, charging energy is negligible, and the spectrum is insensitive to V_g . The ground state is given by nearly degenerate states of opposite parity (blue and orange), where the maximum separation ΔE vanishes exponentially in E_J/E_C , whereas the distance to the first excited states increases with $\sqrt{E_J E_C}$. As E_J is decreased by decreasing the coupling to the bulk superconductor, the ground-state degeneracy is lifted in the intermediate regime. Here, varying the gate voltage distinguishes the parity states through their differential capacitance $C = \partial\langle\hat{N}\rangle/\partial V_g$, which is larger for the orange parity than for the blue parity. Finally, in the Coulomb regime $E_J \ll E_C$, the spectrum is given by parabolas with well-defined charge number N . If V_g is tuned to a minimum of a charge parabola, the two lowest-energy parity states are separated by E_C , which can be used to impose a certain parity on the island. The intermediate regime can also be understood from the emergence of avoided crossings between charge states of equal parity as E_J is increased.

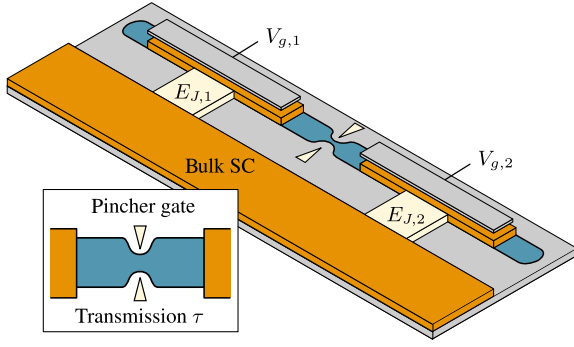


FIG. 16. Two Majorana Cooper pair boxes connected to the same bulk superconductor with Josephson energies $E_{J,1}$ and $E_{J,2}$, and top gate voltages $V_{g,1}$ and $V_{g,2}$, respectively. The islands are connected through the spin-polarized conducting channel, in which the interisland transmission probability τ can be tuned by a pincher gate.

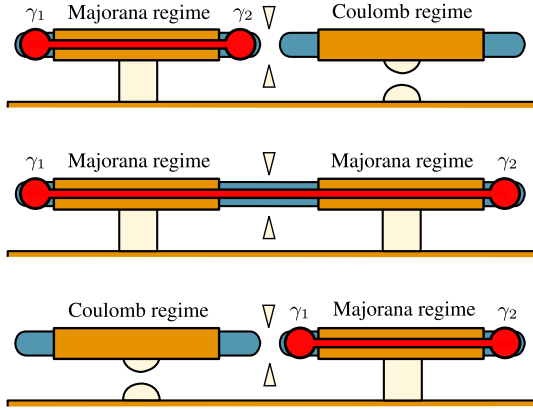


FIG. 17. Two-step protocol for moving Majoranas γ_1 and γ_2 from the left island, initially tuned to the Majorana regime, to the right island, initially tuned to the Coulomb regime with even parity. First, the two islands are coupled by increasing the transmission to $\tau = 1$ and tuning the right island to the Majorana regime, shuttling γ_2 to the right island. The two islands now form a single connected superconducting island with Majoranas γ_1 and γ_2 . In order to move γ_1 to the right island, the transmission is reduced back to $\tau = 0$, and the left island is tuned to the Coulomb regime.

accordingly, so moving the Majorana will not flip the parity state. Finally, γ_1 can be moved to the right island by tuning the left island into the Coulomb regime and then decoupling the two islands.

Majorana Cooper pair boxes arranged in a T junction geometry with three pincher gates between three islands (see Fig. 18) form the basic building block of our proposed network and implement all required moving operations. Opening any pair of pincher gates couples the respective islands. For instance, opening the right and bottom pincher gates in the left configuration of Fig. 18 moves γ_3 from the

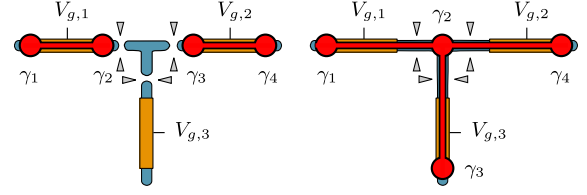


FIG. 18. T-junction geometry consisting of three mesoscopic superconducting islands coupled through a three-terminal junction involving three pincher gates. Left diagram: If all three pincher gates are closed, the islands are decoupled and host a pair of Majoranas each. In dispersive read-out, tuning $V_{g,1}$ and $V_{g,2}$ is used to measure the parities $i\gamma_1\gamma_2$ and $i\gamma_3\gamma_4$, respectively. Right diagram: Opening the right and bottom pincher gate while tuning the bottom island into the Majorana regime moves γ_3 to the bottom island. Subsequently, opening the left pincher gate connects all three islands, where γ_2 is shared by all islands in the three-terminal junction. Connecting any of the top gates to a resonant circuit allows for dispersive read-out of the total parity $-\gamma_1\gamma_2\gamma_3\gamma_4$.

right island to the bottom island. Opening the remaining pincher gate connects the left island to the other two superconductors, such that γ_2 becomes a Majorana shared by all three islands.

B. Fermion-parity measurements

The set of required operations is completed by measurements of the fermion parity of $2n$ Majoranas. The fermion parity $i\gamma_1\gamma_2$ of an island can be measured by tuning to the Coulomb regime and measuring the charge on the island (parity-to-charge conversion). In an alternative scheme, the superconducting island is tuned into the intermediate regime $E_J \sim E_C$ with the gate voltage set such that N_0 is, say, an even number. Then, the even-parity state is at the minimum of a charge parabola, while the odd-parity state sits at an avoided crossing between two charge parabolas. Consequently, the charge on the island is insensitive against variations of the gate voltage in the even-parity state, but it is susceptible in the odd state; i.e., the two parity states differ in the differential capacitance

$$C = \frac{\partial \langle \hat{N} \rangle}{\partial V_g}. \quad (10)$$

When incorporating the island into a resonant circuit, the resonant frequency depends on the differential capacitance. Thus, a measurement of the resonance frequency constitutes a parity measurement, referred to as dispersive read-out [83–86].

This dispersive read-out scheme can be generalized to measuring the fermion parity of $2n$ Majoranas. Moving the $2n$ Majoranas onto one connected superconducting island that is tuned away from the Majorana regime with suitably chosen gate voltage, the parity can be read-out by incorporating this island into a resonant circuit and proceeding

as before. An experimental limitation is set by the decrease of the charging energy with increasing island size. Typically, for nanowire networks, the charging energy decreases linearly with the system size. Therefore, the required precision in the gate voltage control increases linearly with the size of the island. This scheme can, for instance, be applied to measure the four-Majorana parity operator $-\gamma_1\gamma_2\gamma_3\gamma_4$. In the right configuration of the T junction in Fig. 18, all pincher gates are opened and the three islands form one connected mesoscopic superconductor hosting four Majorana zero modes. The spectrum of this effective Majorana Cooper pair box will be the same as in Fig. 15, but with a correspondingly lower charging energy and states differing in *total* parity $(i\gamma_1\gamma_2)(i\gamma_3\gamma_4)$. Thus, dispersive read-out now measures this four-Majorana parity operator.

In this manner, the $2n$ -Majorana parity operator of an island hosting $2n$ Majoranas can be measured via dispersive read-out by connecting any of the top gates to a resonant circuit and measuring the quantum capacitance [83]. Since at least one top gate in each hexagonal cell needs to be connected to its own read-out circuit, incorporating the read-out hardware in a two-dimensional architecture is an experimental challenge. Similar to superconducting qubit platforms, few-qubit quantum computers may allow for an on-chip implementation of both qubit and read-out hardware. However, as each gate in the two-dimensional platform needs to be addressed individually, large-scale quantum computing will require the gates to be contacted to the control hardware via the third dimension, i.e., out of plane. In fact, the integration of the control and measurement hardware in a three-dimensional architecture is the subject of current experimental efforts [87–89] to scale up superconducting qubit platforms.

Alternative schemes that were proposed for the parity read-out of Majorana-based qubits [35,70,71] include charge reflectometry, where a resonator is used to probe the island's energy levels as opposed to the differential capacitance, and charge sensing, which employs the Coulomb regime to read out the average charge on the island. The implementation and benchmarking of different parity read-out techniques is the subject of ongoing experimental efforts.

Having discussed the implementation of the operations required of topological superconductor networks, we now investigate error sources of Majorana Cooper pair boxes and how well they can be corrected by diamond color codes in the following section.

V. FEASIBILITY ESTIMATE

The performance of diamond color-code qubits in topological superconductor networks depends on the error sources of Majorana Cooper pair boxes. Even though the parity states are degenerate in the Majorana regime for $E_J \gg E_C$, a finite overlap between Majorana wave

functions on one island will split the degeneracy. Still, this splitting is exponentially suppressed in the island size. Overlap between Majoranas of neighboring islands can also lead to errors, but the overlap is proportional to the controlled tunneling amplitude between neighboring islands and is thus also exponentially small.

An error that is not necessarily exponentially suppressed occurs when an outside electron tunnels onto an island. This process is called quasiparticle poisoning, which is presumably the dominant error source in Majorana-based qubits. In the following, we model poisoning on any of the two islands encoding a physical qubit by the application of one of the four Majorana operators. This not only changes the total parity sector of the qubit, but it also leads to a logical Pauli error depending on the Majorana involved in the process. The change of the parity sector is inconsequential to the qubit since, in the encodings of both parity sectors in Eqs. (1) and (2), the physical qubit operators are $\sigma_z = i\gamma_1\gamma_2$ and $\sigma_x = i\gamma_2\gamma_3$. Therefore, merely switching the parity sector leaves both the logical information and the logical braid operations $B_{1,2}$ and $B_{2,3}$ unchanged. However, γ_1 anticommutes with σ_z , γ_2 anticommutes with σ_z and σ_x , and γ_3 anticommutes with σ_x . Therefore, poisoning of γ_1 leads to a σ_x error, of γ_2 to a σ_y error, of γ_3 to a σ_z error, and of γ_4 to no error. We discuss this in further detail in Appendix F. Moreover, we discuss more general error sources that are not described by a single Majorana operator.

Since σ_y errors correspond to both a σ_x and a σ_z error, the quasiparticle poisoning time defines a time scale on which σ_x -type and σ_z -type errors occur at equal rates. Current experiments suggest that the quasiparticle poisoning time of mesoscopic superconducting islands might be of the order of milliseconds [41–43], although we point out that these experiments were performed in a regime where the superconducting islands were not floating but connected to a pair of normal-metal leads. We note that even though the regime of equally likely σ_x - and σ_z -type errors is the one considered in the following discussion, this is actually the worst-case scenario for error correction. If one error type is known to occur more often, these errors have been shown to be correctable with fewer resources [90], albeit by changing the code and therefore giving up on transversal gates. But even without abandoning color codes, a biased error source can be taken into account by measuring the corresponding syndrome type more frequently than the other, thereby reducing the code cycle duration and hence the error rate.

In nontopological architectures, random Pauli errors are usually not a realistic error model since relaxation processes from excited states to ground states are not described by unitary operations. For topological hardware, on the other hand, there are no transitions between different parity states that would allow for relaxation from one qubit state to the other. Therefore, we believe that random Pauli errors

should be a reasonable error model for topological physical qubits. With this error model, the physical error threshold for color codes is about 11% [57,62], where the physical error rate is the probability for a physical error on one physical qubit after one code cycle. For our physical hardware, this physical error rate is $p_{\text{phys}} = 1 - e^{-\tau_c/\tau_p} \approx \tau_c/\tau_p$, where τ_p is given by the quasiparticle poisoning time and τ_c is the duration of a code cycle. As moving Majoranas can be done at nanosecond time scales [91] without introducing significant diabatic errors, the code cycle duration is mainly determined by the time required for parity measurement. Dispersive read-out on superconducting qubits suggests that this can be done on microsecond time scales [83,84] or faster. For quasiparticle poisoning times of the order of milliseconds, the physical error rate would be $p_{\text{phys}} \approx 10^{-3}$, which is well below threshold.

In order to estimate the survival time of logical qubits and the performance gain of diamond color codes over triangular color codes, we use a Monte Carlo simulation of the quantum error-correcting code for the aforementioned error model using a lookup table decoder. We note that our decoder does not take correlations between σ_x and σ_z errors into account, which could further enhance the correction procedure with a suitable decoder. A detailed discussion of the simulation is found in Appendix G. In Fig. 19, we show the logical error rate as a function of physical error rate for the first three lowest-distance triangular color codes and the first two diamond color codes. The simulation reproduces the error threshold of about 11% and shows that the logical error rate of diamond color codes is indeed lower compared to a triangular color code of the same code distance. Furthermore, we find that, already for the $d = 5$

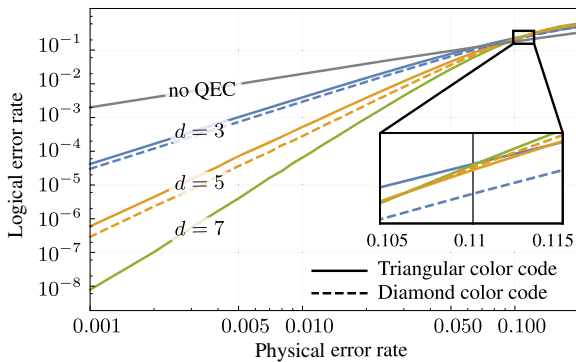


FIG. 19. Logical error rate as a function of physical error rate obtained from Monte Carlo simulation with a lookup table decoder for triangular (solid line) and diamond (dashed line) color codes with code distances $d = 3$ and $d = 5$, and for the triangular color code with $d = 7$. The sample size is between 10^7 and 10^{10} trials for data point corresponding to high and low logical error rates, respectively. The upper line show the logical error rate without quantum error correction. The inset zooms into the crossover region around $p_{\text{phys}} \sim 11\%$.

diamond color code of Fig. 9(b) with $p_{\text{phys}} = 10^{-3}$, the survival time of logical qubits is approximately 35 000 code cycles until the probability for a logical error reaches 1%. In order to determine the survival time for larger code distances, a more efficient decoder needs to be used, such as an iterative decoder [56] or a color clustering decoder [57]. Both slightly lower the error threshold to 7.8% and 9.75%, respectively. Furthermore, a decoder may keep track of multiple rounds of syndrome extraction in order to take measurement errors into account. While it is not known how read-out errors affect the logical error rate, their effect on the error threshold has been studied [62]. As a concrete example, a 95% read-out fidelity lowers the error threshold from about 11% to 2.5%. A numerical study of the corresponding logical error rate would help quantify the performance of color codes, but it goes beyond the scope of this work.

Still, we may extrapolate our results to at least estimate the survival time for higher-distance codes. Details on this are found in Appendix G. The extrapolation suggests that for $p_{\text{phys}} = 10^{-3}$, $\tau_c = 1 \mu\text{s}$, and the more stringent requirement that the logical error probability stays below 10^{-6} , the $d = 19$ diamond color code has a survival time of several years, implying that, with an overhead of roughly 500 physical qubits per logical qubit, quantum computations may run for reasonably long durations. We note that diamond color-code qubits are not resource efficient in the number of physical qubits but only a useful construction for transversal CNOTs and multitarget CNOTs, as discussed in Sec. III. Since equal-distance triangular color codes do not have a substantially higher logical error rate, the number of physical qubits per logical qubit can be reduced by a factor of 2, if data qubits are encoded using triangles instead of diamonds.

VI. CONCLUSION

In this work, we have studied the interplay of topological hardware and topological error-correcting software. Using topological superconductor networks, we have devised a scalable architecture for universal fault-tolerant topological quantum computation, which can be realized with voltage-controlled Majorana Cooper pair boxes as basic building blocks. The underlying physical qubits are hexagonal-cell qubits, which allow for universal quantum computing with topologically protected Clifford gates, fast multitarget CNOT gates, and ancilla-free syndrome read-out. For quantum error correction, we employ topological color codes. Their set of transversal gates coincides with the topologically protected Clifford gates, which enables the logical gates to retain their topological protection due to the topological hardware. This makes color codes a natural fit to Majorana-based hardware, as they seamlessly combine topological hardware with topological software while still benefiting from the topological protection of both. Moreover, color codes also feature a

reduced time overhead for gate operations and a higher error threshold compared to surface codes, even in the presence of measurement errors during stabilizer read-out. In a qubit arrangement consisting of a row of data qubits, a magic-state distillery, and a CNOT bypass, logical single-qubit Clifford gates have a fast transversal implementation, CNOTs between any pair of data qubits have a constant-time overhead, and magic states can be distilled faster using transversal multitarget CNOT gates. Our architecture is not restricted to implementations using Majorana Cooper pair boxes, but it can be applied to any realization of a topological superconductor network, provided that Majoranas can be moved, that their parities can be measured, and that some implementation of a physical T gate is available.

Considering the particular geometry of a Majorana-based color-code quantum computer presented in this work, i.e., hexagonal-cell qubits and 6.6.6 diamond color codes, we make no claim of this geometry being optimal in terms of space and time overhead. Studies of different network layouts and color-code schemes may reduce the overhead. Still, it is not clear how different code layouts and decoders affect the logical error rate. In particular, 4.8.8 color codes require fewer physical qubits compared to 6.6.6 codes with the same code distance. However, as we show in Appendix H, they also feature a higher logical error rate, even though they have the same code distance and error threshold. Similarly, for the comparison of triangular and diamond codes, neither code distance nor error threshold is a predictive figure of merit for logical error rates. We therefore encourage studies of the logical error rate of topological codes, in order to quantify the performance of codes beyond the already well-studied error thresholds and code distances. Moreover, in order to further quantify the performance of a topological color-code quantum computer, it would be interesting to estimate the number of code cycles required for actual computational tasks in an arrangement of data qubits and magic-state distilleries.

On the hardware side, the past years have shown considerable experimental progress towards the realization of Majorana zero modes through the interplay of superconductivity, spin-orbit coupling, and one-dimensional spin-polarized channels. This work is expected to provide further motivation for ongoing efforts to achieve braiding of Majoranas in these systems. On a more general note, aiming at merging ideas of both hardware- and software-based topological protection, we hope that our work further stimulates research efforts bringing the fields of condensed-matter physics and quantum information theory closer together.

ACKNOWLEDGMENTS

We thank F. Pastawski, S. Acero, A. Bauer, M. Filippone, M. Geier, M. Gluza, A. Haim, T. Karzig, Y. Oreg, D. Sabonis, Y. Vinkler-Aviv, and other participants of

the CRC 183 for illuminating discussions. This work has been supported by the DFG (CRC 183), the ERC (TAQ), the Templeton Foundation, and the EC (AQuS).

APPENDIX A: PROTOCOL FOR A TRANSVERSAL CNOT GATE

In Fig. 20, we show that in hexagonal-cell qubits, CNOT gates between control and target qubits arranged on a line can be performed simultaneously. This is a generalization of the protocol introduced in Fig. 5 using the quantum circuit in Fig. 4. Since in diamond color codes, control and target qubits are also arranged on a line, all transversal CNOT gates in color codes can be performed simultaneously.

APPENDIX B: MULTITARGET CNOT BY PARITY MEASUREMENT

Here, we show that a multitarget CNOT gate can be realized using Clifford gates and qubit parity measurements (see Fig. 6). The multitarget CNOT operator

$$\text{CNOT}_n = |0\rangle\langle 0| \otimes \mathbb{1}^{\otimes n} + |1\rangle\langle 1| \otimes \sigma_x^{\otimes n} \quad (\text{B1})$$

flips all n target qubits if the control qubit is in the $|1\rangle$ state. An n -qubit parity measurement with outcome $m = 0$ for even and $m = 1$ for odd parity is equivalent to an operation

$$P_z = \frac{1}{2} [\mathbb{1}^{\otimes n} + (-1)^m \sigma_z^{\otimes n}]. \quad (\text{B2})$$

Similarly, an n -qubit parity measurement in the σ_x basis is

$$P_x = H^{\otimes n} P_z H^{\otimes n} = \frac{1}{2} [\mathbb{1}^{\otimes n} + (-1)^m \sigma_x^{\otimes n}]. \quad (\text{B3})$$

Thus, the circuit in Fig. 6 in the basis $|c\rangle \otimes |a\rangle \otimes |t\rangle^{\otimes n}$, where c , a , and t denote the control, ancilla, and the n target qubits, respectively, is

$$\begin{aligned} U = & \left(\mathbb{1} \otimes \frac{1}{2} [\mathbb{1} + (-1)^{m_3} \sigma_z] \otimes \mathbb{1}^{\otimes n} \right) \\ & \times \left(\mathbb{1} \otimes \frac{1}{2} [\mathbb{1}^{\otimes n+1} + (-1)^{m_2} \sigma_x^{\otimes n+1}] \right) \\ & \times \left(\frac{1}{2} [\mathbb{1}^{\otimes 2} + (-1)^{m_1} \sigma_z^{\otimes 2}] \otimes \mathbb{1}^{\otimes n} \right) \\ & \times (\mathbb{1} \otimes |+\rangle\langle +| \otimes \mathbb{1}^{\otimes n}), \end{aligned} \quad (\text{B4})$$

where the final correction is not yet applied. Tracing out the ancilla qubit yields

$$\begin{aligned} U = & \frac{1}{2} (\mathbb{1} + (-1)^{m_1+m_3} \sigma_z) \otimes \mathbb{1}^{\otimes n} \\ & + \frac{1}{2} (-1)^{m_2} (\mathbb{1} - (-1)^{m_1+m_3} \sigma_z) \otimes \sigma_x^{\otimes n}. \end{aligned} \quad (\text{B5})$$

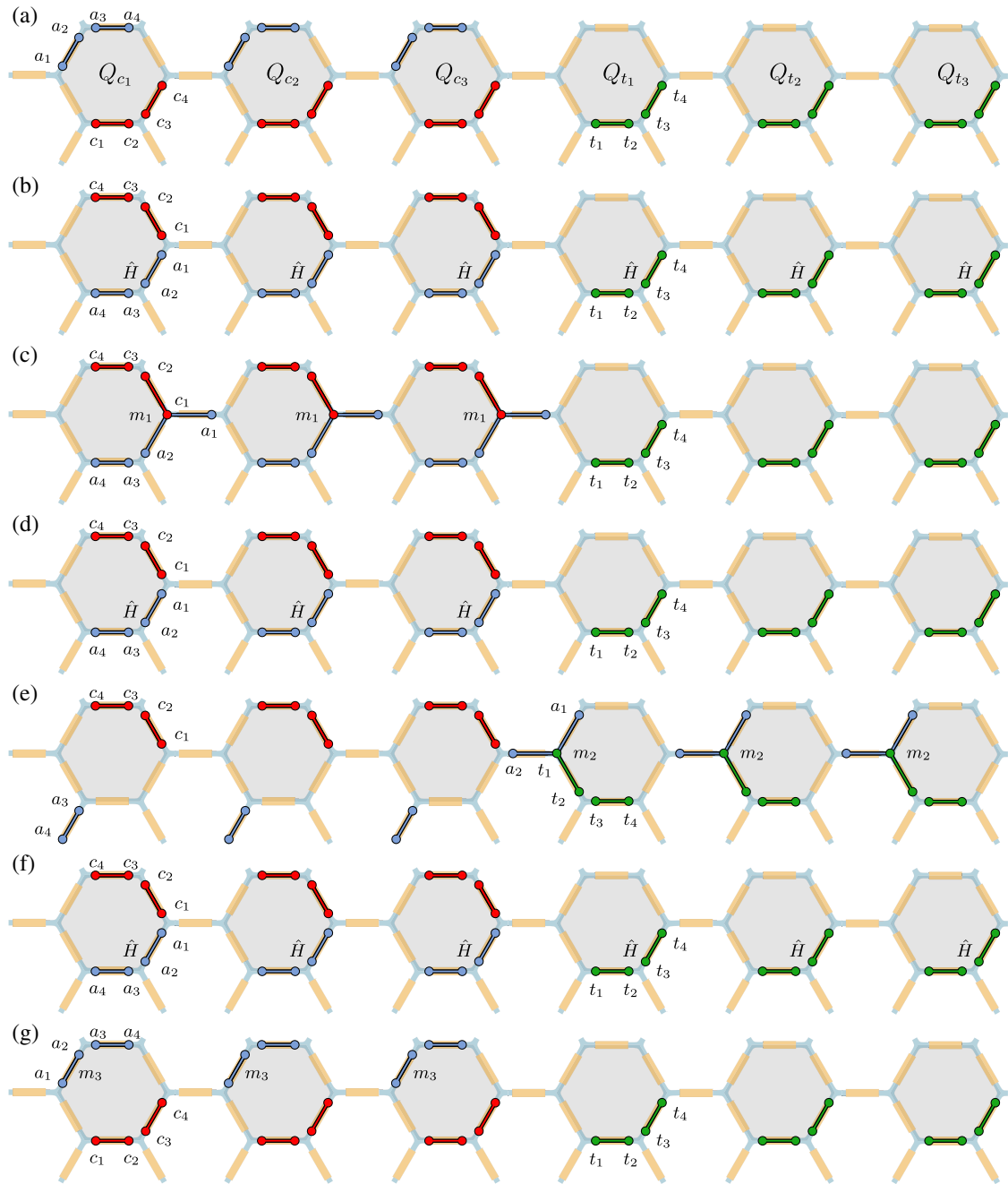


FIG. 20. Protocol for three simultaneous CNOTs between three control qubits $Q_{c_1-c_3}$ and three target qubits $Q_{t_1-t_3}$ using the quantum circuit in Fig. 4. In the cell occupied by the control qubits (red), ancillas (blue) are initialized in the $|0\rangle$ state (a) and moved to the double T junction of the cell for a Hadamard gate (b). The first two Majoranas of the control and ancilla qubits are moved onto a connected island, and the four-Majorana parity is measured (c), corresponding to a two-qubit parity measurement with outcome m_1 . The ancillas are moved back to the double T junction for another H gate (d). The third and fourth Majoranas a_3 and a_4 of each ancilla qubit are moved into the lower leg of their hexagonal cell, such that the remaining ancilla Majoranas can move to the target qubit cells for a four-Majorana parity measurement (e). The ancilla Majoranas are then moved back to the control cells for an H gate (f). Finally, all qubits return to their initial positions (g), and the ancilla qubits are measured by measuring the two-Majorana parity m_3 .

Depending on the measurement outcomes $m_1 + m_3$ and m_2 , there are four possible uncorrected operations $U_{m_1+m_3, m_2}$ (see Table I). Thus, after the correction $\sigma_z^{m_2} \otimes (\sigma_x^{m_1+m_3})^{\otimes n}$, the

circuit in Fig. 6 precisely yields the multitarget CNOT gate CNOT_n using only three measurements, as opposed to $3n$ measurements for n individual CNOTs.

TABLE I. Uncorrected gate $U_{m_1+m_3, m_2}$ and necessary correction based on measurement outcomes m_1 , m_2 , and m_3 .

$m_1 + m_3$	m_2	$U_{m_1+m_3, m_2}$	Correction
0	0	$ 0\rangle\langle 0 \otimes \mathbb{1}^{\otimes n} + 1\rangle\langle 1 \otimes \sigma_x^{\otimes n}$	$1 \otimes \mathbb{1}^{\otimes n}$
0	1	$ 0\rangle\langle 0 \otimes \mathbb{1}^{\otimes n} - 1\rangle\langle 1 \otimes \sigma_x^{\otimes n}$	$\sigma_z \otimes \mathbb{1}^{\otimes n}$
1	0	$ 1\rangle\langle 1 \otimes \mathbb{1}^{\otimes n} + 0\rangle\langle 0 \otimes \sigma_x^{\otimes n}$	$1 \otimes \sigma_x^{\otimes n}$
1	1	$ 1\rangle\langle 1 \otimes \mathbb{1}^{\otimes n} - 0\rangle\langle 0 \otimes \sigma_x^{\otimes n}$	$\sigma_z \otimes \sigma_x^{\otimes n}$

APPENDIX C: DETAILS ON LATTICE SURGERY

The stabilizers that are measured during lattice surgery [34] are shown in Fig. 21. In the following, we describe the protocol for ZZ-parity measurement, but this can be straightforwardly used for XX-parity measurement by simply swapping $Z \leftrightarrow X$ in the protocol. As one can verify by direct inspection, in this ZZ-parity measurement between two distance d codes, $\lceil d/2 \rceil$ new Z stabilizers are introduced and $2\lfloor d/2 \rfloor$ X stabilizers are replaced by $\lfloor d/2 \rfloor$ octagon stabilizers. Since d is always odd, exactly one new stabilizer is introduced. This reduces the number of logically encoded qubits by one, implying that in this process, one bit of information is measured. In the following, we would like to make the case that the measured bit is precisely the ZZ parity.

First, notice that in the absence of errors, the extended octagon stabilizers will not detect anyons (i.e., have a measurement outcome +1) since they are products of preexisting stabilizers. In this error-free setting, the only stabilizers that give a nontrivial measurement outcome are the $2\lfloor d/2 \rfloor$ newly created Z stabilizers. The product of these stabilizers is exactly the logical two-qubit parity $(\sigma_z \sigma_z)_L$. Another way to understand this process is in terms of anyons. If in the error-free setting an anyon is detected

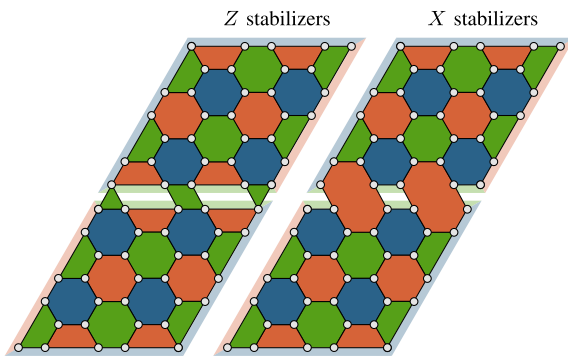


FIG. 21. Stabilizers that are measured to obtain the ZZ parity between two logical qubits using lattice surgery. In contrast to usual color-code stabilizer measurements, lattice surgery requires measurements where the support of X and Z stabilizers does not coincide. The left panel shows the required Z stabilizers, and the right panel the X stabilizers, which differ along the shared boundary. To obtain the XX parity between two qubits, one simply has to swap $Z \leftrightarrow X$ in the protocol.

on one of the new Z stabilizers (depicted as green in Fig. 21), this means that the number of strings from the upper code terminating on said plaquette differs in parity from the number of strings coming from the lower code. Thus, the total parity of all newly created plaquettes measures exactly the difference of strings from the upper and lower codes, which is precisely the two-qubit parity.

Lastly, in order to convince oneself of the fault tolerance of the above process, it is sufficient to check that strings corresponding to logical operations in the new settings still involve at least d physical qubits. In Fig. 21, this has to be fulfilled for logical $(\sigma_x)_L$, $(\sigma_z)_L$, and $(\sigma_x \sigma_x)_L$ operations but not for $(\sigma_z \sigma_z)_L$ since this commutes with the parity measurement.

APPENDIX D: DETAILS OF COLOR-CODE STATE INJECTION

Figure 12 shows the protocol for the injection of a single qubit state $|\psi\rangle$ into a logical state $|\psi_L\rangle$ encoded in a diamond color code. This is a direct adaptation of the protocol in Ref. [34], where this protocol was introduced for the specific case of triangular 4.8.8 color codes. Here, we explain how this protocol achieves the state injection adapted to our situation.

The left panel of Fig. 12 depicts the stabilizers measured before state injection. The number of stabilizers is exactly the same as the number of physical qubits, and thus no logical qubits can be encoded. Since all stabilizers are measured and errors are corrected for, no anyonic excitations are present initially. The fact that there are only two boundaries implies that an even number of strings has to leave each boundary. This is a consequence of errors always creating pairs of anyons of the same color or triples of all three colors (and combinations thereof), and of the fact that boundaries can only host anyons of their respective color.

In the concrete example shown in Fig. 12, the Z parity of all qubits along the blue boundary is even, $\sigma_z^{\otimes n} = +1$. The same holds for the X parity and equivalent measurements along the red boundary. Importantly, this statement generalizes and holds for all color codes with two boundaries, regardless of code distance, geometry, and tiling.

To inject the state of the single physical qubit into the color code, the stabilizers shown in the right panel of Fig. 12 are measured. Even if no errors on physical qubits occur, the new red plaquettes might still host anyons. Importantly, they are not corrected according to the most likely error configuration producing this syndrome; instead, they will manifest themselves in the syndrome read-out and can be corrected.

The blue boundary after state injection differs only by the addition of the new physical qubit. Thus, measurements of the logical state along this boundary are given by the state of the new physical qubit alone. The way in which

anyons on the new red plaquettes are corrected ensures that the same holds for all other measurements of logical operations as well. This proves that the protocol successfully injects the state of the single physical qubit $|\psi\rangle$ into the logical state $|\psi_L\rangle$ encoded in the color code.

APPENDIX E: CONSTANT-TIME OVERHEAD CNOT

The quantum circuit in Fig. 22 describes the constant-time overhead CNOT protocol in Fig. 10(c). The protocol involves a control qubit $|c\rangle$, a target qubit $|t\rangle$, and three $|+\rangle$ ancillas, where the third ancilla may be thought of as the ancilla that is part of the CNOT protocol of Fig. 4. The ZZ parity between this third ancilla and the control qubit is not measured directly but as the sum of the first three parity measurements in the circuit $m_1 + m_2 + m_3$. The XX parity between the third ancilla and the target is measured with outcome m_4 , and the third ancilla is read-out with outcome m_7 . This is the reason for the $\sigma_x^{m_1+m_2+m_3+m_7}$ correctional operation on the target and the $\sigma_z^{m_4}$ correction on the control qubit. However, these operations alone leave the first two ancilla qubits entangled with the control qubit in a state of the type $|\psi\rangle = \alpha|0,0,0\rangle + \beta|1,1,1\rangle$. In order to safely discard the two ancilla qubits without affecting the control qubit, they are measured in the X basis with outcomes m_5 and m_6 , leading to a $\sigma_z^{m_5+m_6}$ correction on the control qubit.

APPENDIX F: QUASIPARTICLE POISONING

In the following, we discuss qubit errors due to quasiparticle poisoning. We find that merely changing the parity sector of an island pair is inconsequential to the qubit, and we consider more general error sources that are not described by the processes discussed in the main text. We define the three Pauli operators in the space of even and odd-parity states $\{|e\rangle, |o\rangle\}$ of a topological superconducting island,

$$\tau_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \tau_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \tau_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \quad (\text{F1})$$

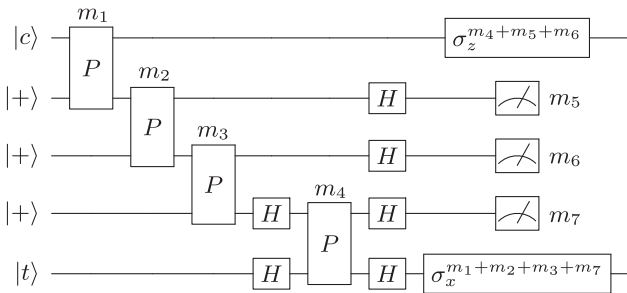


FIG. 22. Quantum circuit corresponding to the constant-time overhead CNOT gate in Fig. 10(c).

The four Majorana operators $\gamma_1, \dots, \gamma_4$ of an island pair can be represented in terms of these Pauli operators as

$$\begin{aligned} \gamma_1 &= \tau_x \otimes \mathbb{1}, & \gamma_2 &= -\tau_y \otimes \mathbb{1}, \\ \gamma_3 &= \tau_z \otimes \tau_x, & \gamma_4 &= -\tau_z \otimes \tau_y, \end{aligned} \quad (\text{F2})$$

upon choosing a specific order of modes and by invoking the Jordan-Wigner transformation. These operators are Hermitian $\gamma_i = \gamma_i^\dagger$ and fulfill the anticommutation relations $\{\gamma_i, \gamma_j\} = 2\delta_{i,j}$. Our two qubit encodings are

$$|0_e\rangle = |e\rangle \otimes |e\rangle, \quad |1_e\rangle = |o\rangle \otimes |o\rangle \quad (\text{F3})$$

in the even-parity sector and

$$|0_o\rangle = |e\rangle \otimes |o\rangle, \quad |1_o\rangle = |o\rangle \otimes |e\rangle \quad (\text{F4})$$

in the odd-parity sector. Therefore, in both parity sectors, the logical qubit operators are $\sigma_z = i\gamma_1\gamma_2$ and $\sigma_x = i\gamma_2\gamma_3$. Consider a quasiparticle poisoning event described by the application of γ_1 . The operator γ_1 maps $|0_e\rangle \leftrightarrow |1_o\rangle$ and $|1_e\rangle \leftrightarrow |0_o\rangle$; i.e., it switches the parity sector and applies a logical σ_x operation. Similarly, Eq. (F2) implies that γ_2 applies a logical σ_y operation and γ_3 a logical σ_z operation. The operator γ_4 only switches the parity sector without changing the logical information. This is not surprising since it is the only Majorana operator that is not part of either σ_z or σ_x . What is more, invoking the fermion-parity superselection rule, it is clear that the specific order of modes used in this argument is not relevant, i.e., that the specific Jordan-Wigner string plays no role.

To further demonstrate that, in general, the information about the parity sector is irrelevant for quantum computation, we write the state of an island pair as a product of the qubit state and the parity state. An island pair is a four-level system with the four basis states in Eqs. (F3) and (F4). Instead of describing these states in terms of the fermion parities of the first and second islands, we can transform the basis to a product of an eigenstate $\{|0\rangle, |1\rangle\}$ of the qubit operator $\sigma_z = i\gamma_1\gamma_2$ (qubit state) and an eigenstate $\{|p_e\rangle, |p_o\rangle\}$ of the total parity operator $p = -\gamma_1\gamma_2\gamma_3\gamma_4$ (parity state). In this basis, the four states are

$$\begin{aligned} |0_e\rangle &= |0\rangle \otimes |p_e\rangle, & |1_e\rangle &= |1\rangle \otimes |p_e\rangle, \\ |0_o\rangle &= |0\rangle \otimes |p_o\rangle, & |1_o\rangle &= |1\rangle \otimes |p_o\rangle. \end{aligned} \quad (\text{F5})$$

Incidentally, the transformation matrix that maps from Eqs. (F3) and (F4) to Eq. (F5) is a CNOT. Braiding operations and measurements of the qubit only affect the qubit state but not the parity state since they are comprised of operators that are products of $\gamma_1\gamma_2$ or $\gamma_2\gamma_3$ and therefore commute with the parity operator p . After this mapping, the poisoning processes that we considered previously (i.e., the application of a Majorana operator) can be written as a

product of operations on the qubit state and on the parity state,

$$\begin{aligned}\gamma_1 &= \tilde{\sigma}_x \otimes \tilde{\tau}_x, & \gamma_2 &= -\tilde{\sigma}_y \otimes \tilde{\tau}_x, \\ \gamma_3 &= \tilde{\sigma}_z \otimes \tilde{\tau}_x, & \gamma_4 &= -\mathbb{1} \otimes \tilde{\tau}_y,\end{aligned}\quad (\text{F6})$$

where $\tilde{\sigma}_i$ and $\tilde{\tau}_i$ are Pauli operators acting on the qubit and parity spaces, respectively. Therefore, these operations do not entangle the qubit with the parity d.o.f. Furthermore, any Jordan-Wigner string $\tau_z \otimes \tau_z$ associated with the operators in Eq. (F2) is mapped onto $\mathbb{1} \otimes \tilde{\tau}_z$, which acts trivially on the qubit state. However, a general operation can, in principle, generate such an entangled state. Thus, the most general description of the entire state of the system is a sum over all 2^n possible $2n$ -island parity sectors,

$$|\psi\rangle = \sum_{\text{parity sectors } p} |\psi_p\rangle \otimes |p\rangle, \quad (\text{F7})$$

where $|p\rangle$ contains all fermion parities of the n island pairs, and $|\psi_p\rangle$ is an n -qubit state. But in our encoding, $|p\rangle$ carries no information relevant to quantum computation. Tracing out the parity state leaves the qubit state in a statistical ensemble. Therefore, the parity d.o.f. acts like an environment to which error sources can couple. Moreover, in the absence of logical errors, different qubit states yield different syndromes after stabilizer read-out. Thus, measuring the syndrome breaks the entanglement between the qubit state and parity state.

Error sources that entangle the qubit with the parity state are not described by products of Majorana operators but by sums of products. Such errors are, in principle, allowed and lead to a nonunitary evolution of the qubit state. These errors are still correctable but are not necessarily described by the error model of random Pauli errors. One example for such an effectively nonunitary process is swapping the parities of two islands that belong to two different qubits, as this entangles the qubit and parity degrees of freedom.

APPENDIX G: MONTE CARLO SIMULATION OF THE DIAMOND COLOR CODE

In order to study the performance gain of low-distance diamond color codes over triangular color codes and estimate the logical error rate for higher-distance codes, we sample the logical error rate in a Monte Carlo simulation. The physical error rate is the probability for at least one error event in a code cycle,

$$p_{\text{phys}} = 1 - \lim_{N \rightarrow \infty} \left(1 - \frac{1}{N} \frac{\tau_c}{\tau_p}\right)^N = 1 - e^{-\tau_c/\tau_p}, \quad (\text{G1})$$

where τ_c is the duration of a code cycle and τ_p is the characteristic time scale on which bit flips and phase flips occur. A physical bit flip or phase flip only occurs at the

end of a code cycle if the bit is flipped an odd number of times within a cycle. The probability of a physical bit flip or phase flip can be calculated from the probability of an odd number of successes in n discrete trials with success probability p , which is

$$p_{\text{odd}} = \frac{1 - (1 - 2p)^n}{2}. \quad (\text{G2})$$

Thus, the physical bit-flip (and phase-flip) probability is

$$\begin{aligned}p_{\text{flip}} &= \lim_{N \rightarrow \infty} \frac{1}{2} \left(1 - \left(1 - \frac{2}{N} \frac{\tau_c}{\tau_p}\right)^N\right) \\ &= \frac{1}{2} (1 - e^{-2\tau_c/\tau_p}) = p_{\text{phys}} - \frac{1}{2} p_{\text{phys}}^2.\end{aligned}\quad (\text{G3})$$

We define the logical error rate p_{log} as the probability for a logical bit flip (or phase flip). Without quantum error correction, $p_{\text{log}} \neq p_{\text{phys}}$ since the absence of a logical error requires the absence of both σ_x and σ_z errors. Thus, the physical qubit needs to pass two trials, and the logical error rate is

$$p_{\text{log}} = 1 - (1 - p_{\text{flip}})^2. \quad (\text{G4})$$

To calculate p_{log} with error correction, we sample through error configurations with a bit-flip probability p_{flip} on each physical qubit, attempt to correct the error using a decoder, and count the number of failure events. The logical error rate is

$$p_{\text{log}} = 1 - \left(1 - \frac{\text{fails}}{\text{trials}}\right)^2. \quad (\text{G5})$$

Our decoder is a pregenerated lookup table, which, given an error syndrome, returns the most likely corresponding error configuration. Since this is not efficient for higher-distance codes, we only simulate the triangular color codes with distances $d = 3$, $d = 5$, and $d = 7$, and diamond color codes with distances $d = 3$ and $d = 5$. On a log-log plot, the logical error rate is linear for low physical error rates (see Fig. 19). The slopes and offsets of these linear functions both grow approximately linearly with increasing code distance, allowing for a rough estimate of the low-error behavior of higher-distance codes through extrapolation.

The survival time of a logical qubit until the probability of a logical error p_{err} reaches a target accuracy p_{target} is

$$\tau_{\text{survival}} = \frac{\ln(1 - p_{\text{target}})}{\ln(1 - p_{\text{log}})}, \quad (\text{G6})$$

where τ_{survival} is the survival time as a number of code cycles. In Fig. 23, we plot the survival time for $p_{\text{target}} = 10^{-6}$ for triangular and diamond color codes obtained from

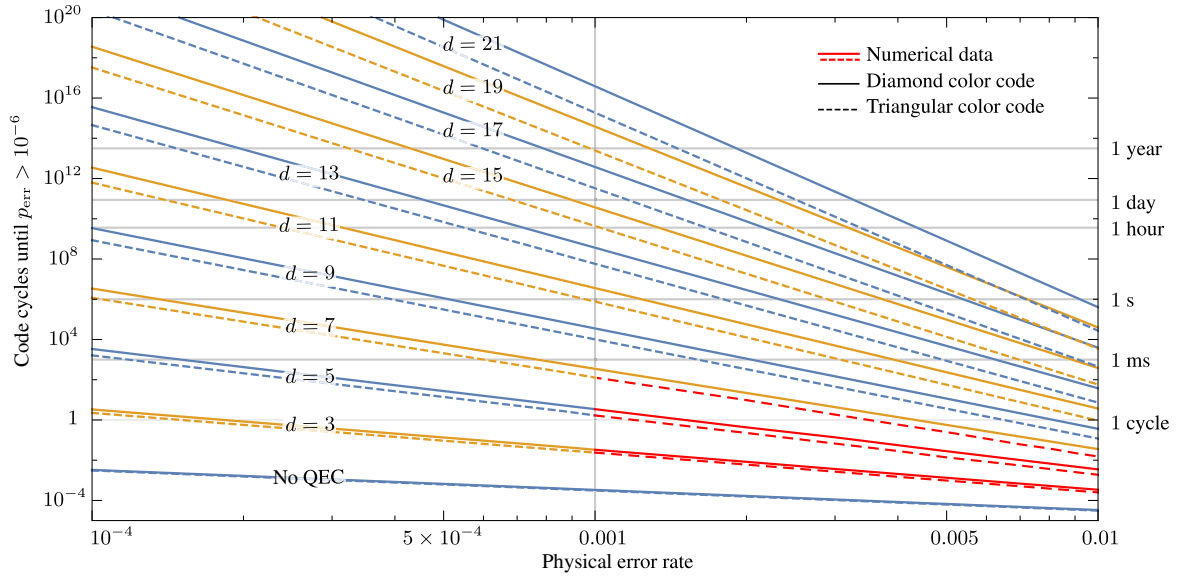


FIG. 23. Survival time of triangular (dashed lines) and diamond (solid lines) color-code qubits until the logical error probability p_{err} reaches 10^{-6} for increasing code distances obtained from extrapolation of the numerical data (red). The labels on the right-hand side of the time axis show the survival time for a code cycle duration of $1 \mu\text{s}$.

numerical data and extrapolation thereof. The extrapolation indicates that for $p_{\text{phys}} = 10^{-3}$ and a code cycle duration $\tau_c = 1 \mu\text{s}$, the survival time of a $d = 19$ diamond color code is of the order of several years.

APPENDIX H: 4.8.8 VS 6.6.6 COLOR CODES

Quantum error-correcting codes are usually classified using their code distances and error thresholds. For triangular and diamond color codes, we have seen that two 6.6.6 color codes with equal code distances can exhibit different logical error rates. Here, we show that neither the code distance nor the error threshold is a predictive figure

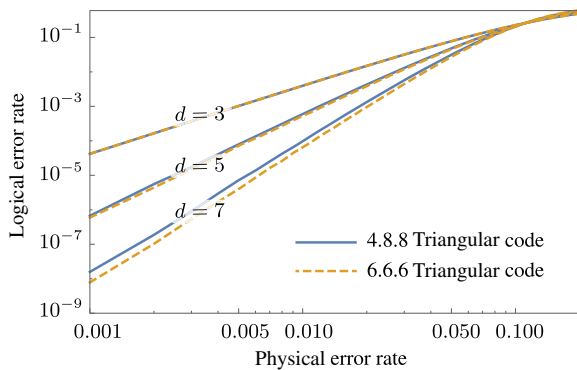


FIG. 24. Logical error rate of the first three 4.8.8 and 6.6.6 triangular color codes. Even though the codes have the same code distance and error threshold, 4.8.8 color codes have a higher logical error rate but require fewer physical qubits per logical qubit.

of merit for the logical error rate, which determines the performance of a code.

In this work, we considered color codes that are defined on lattices with 6.6.6 tiling. A different tiling that allows for color codes is the 4.8.8 tiling, with two types of eight-qubit stabilizers and one type of four-qubit stabilizers. These 4.8.8 codes are considered to be more efficient since triangular 4.8.8 codes require fewer physical qubits per logical qubit compared to the triangular 6.6.6 code with the same code distance.

Figure 24 shows the logical error rate of 4.8.8 and 6.6.6 codes obtained from the previously described Monte Carlo simulation. It shows that the lower physical overhead of 4.8.8 codes comes at the price of a higher logical error rate. Therefore, for a target logical error rate, it is difficult to estimate which code one should use to minimize the physical overhead. To our knowledge, even though code distances and error thresholds of codes are well studied, logical error rates have attracted less attention so far and require further research.

[1] A. Kitaev, *Fault-Tolerant Quantum Computation by Anyons*, *Ann. Phys. (Amsterdam)* **303**, 2 (2003).
 [2] C. Nayak, S. H. Simon, A. Stern, M. Freedman, and S. Das Sarma, *Non-Abelian Anyons and Topological Quantum Computation*, *Rev. Mod. Phys.* **80**, 1083 (2008).
 [3] B. M. Terhal, *Quantum Error Correction for Quantum Memories*, *Rev. Mod. Phys.* **87**, 307 (2015).
 [4] L. Fu and C. L. Kane, *Superconducting Proximity Effect and Majorana Fermions at the Surface of a Topological Insulator*, *Phys. Rev. Lett.* **100**, 096407 (2008).

- [5] Y. Oreg, G. Refael, and F. von Oppen, *Helical Liquids and Majorana Bound States in Quantum Wires*, *Phys. Rev. Lett.* **105**, 177002 (2010).
- [6] R. M. Lutchyn, J. D. Sau, and S. Das Sarma, *Majorana Fermions and a Topological Phase Transition in Semiconductor-Superconductor Heterostructures*, *Phys. Rev. Lett.* **105**, 077001 (2010).
- [7] J. Alicea, *New Directions in the Pursuit of Majorana Fermions in Solid State Systems*, *Rep. Prog. Phys.* **75**, 076501 (2012).
- [8] M. Leijnse and K. Flensberg, *Introduction to Topological Superconductivity and Majorana Fermions*, *Semicond. Sci. Technol.* **27**, 124003 (2012).
- [9] C. W. J. Beenakker, *Search for Majorana Fermions in Superconductors*, *Annu. Rev. Condens. Matter Phys.* **4**, 113 (2013).
- [10] A. Y. Kitaev, *Unpaired Majorana Fermions in Quantum Wires*, *Sov. Phys. Usp.* **44**, 131 (2001).
- [11] V. Mourik, K. Zuo, S. M. Frolov, S. R. Plissard, E. P. a. M. Bakkers, and L. P. Kouwenhoven, *Signatures of Majorana Fermions in in Hybrid Superconductor-Semiconductor Nanowire Devices*, *Science* **336**, 1003 (2012).
- [12] A. Das, Y. Ronen, Y. Most, Y. Oreg, M. Heiblum, and H. Shtrikman, *Zero-Bias Peaks and Splitting in an Al-InAs Nanowire Topological Superconductor as a Signature of Majorana Fermions*, *Nat. Phys.* **8**, 887 (2012).
- [13] S. M. Albrecht, A. P. Higginbotham, M. Madsen, F. Kuemmeth, T. S. Jespersen, J. Nyg, P. Krogstrup, and C. M. Marcus, *Exponential Protection of Zero Modes in Majorana Islands*, *Nature (London)* **531**, 206 (2016).
- [14] M. Deng, S. Vaitiekėnas, E. Hansen, J. Danon, M. Leijnse, K. Flensberg, J. Nygård, P. Krogstrup, and C. Marcus, *Majorana Bound State in a Coupled Quantum-Dot Hybrid-Nanowire System*, *Science* **354**, 1557 (2016).
- [15] H. J. Suominen, M. Kjaergaard, A. R. Hamilton, J. Shabani, C. J. Palmstrøm, C. M. Marcus, and F. Nichele, *Scalable Majorana Devices*, [arXiv:1703.03699](https://arxiv.org/abs/1703.03699).
- [16] J. Preskill, *Reliable Quantum Computers*, *Proc. R. Soc. A* **454**, 385 (1998).
- [17] B. M. Terhal, F. Hassler, and D. P. DiVincenzo, *From Majorana Fermions to Topological Order*, *Phys. Rev. Lett.* **108**, 260504 (2012).
- [18] S. Vijay, T. H. Hsieh, and L. Fu, *Majorana Fermion Surface Code for Universal Quantum Computation*, *Phys. Rev. X* **5**, 041038 (2015).
- [19] S. Vijay and L. Fu, *Physical Implementation of a Majorana Fermion Surface Code for Fault-Tolerant Quantum Computation*, *Phys. Scr. T* **T168**, 014002 (2016).
- [20] L. A. Landau, S. Plugge, E. Sela, A. Altland, S. M. Albrecht, and R. Egger, *Towards Realistic Implementations of a Majorana Surface Code*, *Phys. Rev. Lett.* **116**, 050501 (2016).
- [21] S. Plugge, L. A. Landau, E. Sela, A. Altland, K. Flensberg, and R. Egger, *Roadmap to Majorana Surface Codes*, *Phys. Rev. B* **94**, 174514 (2016).
- [22] B. Eastin and E. Knill, *Restrictions on Transversal Encoded Quantum Gate Sets*, *Phys. Rev. Lett.* **102**, 110502 (2009).
- [23] While this is the notion of transversal gates that we use in this paper, the general definition of transversal gates includes all operations that are tensor products of operations on local subsystems.
- [24] P. Bonderson and C. Nayak, *Quasi-Topological Phases of Matter and Topological Protection*, *Phys. Rev. B* **87**, 195451 (2013).
- [25] H. Bombin, G. Duclos-Cianci, and D. Poulin, *Universal Topological Phase of Two-Dimensional Stabilizer Codes*, *New J. Phys.* **14**, 073048 (2012).
- [26] H. Bombin and M. A. Martin-Delgado, *Topological Quantum Distillation*, *Phys. Rev. Lett.* **97**, 180501 (2006).
- [27] A. J. Landahl, J. T. Anderson, and P. R. Rice, *Fault-Tolerant Quantum Computing with Color Codes*, [arXiv:1108.5738](https://arxiv.org/abs/1108.5738).
- [28] T. Karzig, Y. Oreg, G. Refael, and M. H. Freedman, *Universal Geometric Path to a Robust Majorana Magic Gate*, *Phys. Rev. X* **6**, 031019 (2016).
- [29] M. Barkeshli and J. D. Sau, *Physical Architecture for a Universal Topological Quantum Computer Based on a Network of Majorana Nanowires*, [arXiv:1509.07135](https://arxiv.org/abs/1509.07135).
- [30] S. Bravyi and A. Kitaev, *Universal Quantum Computation with Ideal Clifford Gates and Noisy Ancillas*, *Phys. Rev. A* **71**, 022316 (2005).
- [31] S. Bravyi, B. M. Terhal, and B. Leemhuis, *Majorana Fermion Codes*, *New J. Phys.* **12**, 083039 (2010).
- [32] S. Vijay and L. Fu, *Quantum Error Correction for Complex and Majorana Fermion Qubits*, [arXiv:1703.00459](https://arxiv.org/abs/1703.00459).
- [33] M. B. Hastings, *Small Majorana fermion codes*, [arXiv:1703.00612](https://arxiv.org/abs/1703.00612).
- [34] A. J. Landahl and C. Ryan-Anderson, *Quantum Computing by Color-Code Lattice Surgery*, [arXiv:1407.5103](https://arxiv.org/abs/1407.5103).
- [35] D. Aasen, M. Hell, R. V. Mishmash, A. Higginbotham, J. Danon, M. Leijnse, T. S. Jespersen, J. A. Folk, C. M. Marcus, K. Flensberg, and J. Alicea, *Milestones Toward Majorana-Based Quantum Computing*, *Phys. Rev. X* **6**, 031016 (2016).
- [36] V. Bouchiat, D. Vion, P. Joyez, D. Esteve, and M. H. Devoret, *Quantum Coherence with a Single Cooper Pair*, *Phys. Scr. T* **T76**, 165 (1998).
- [37] 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*, *Phys. Rev. A* **76**, 042319 (2007).
- [38] F. Hassler, A. R. Akhmerov, and C. W. J. Beenakker, *The Top-Transmon: A Hybrid Superconducting Qubit for Parity-Protected Quantum Computation*, *New J. Phys.* **13**, 095004 (2011).
- [39] B. Van Heck, A. Akhmerov, F. Hassler, M. Burrello, and C. Beenakker, *Coulomb-Assisted Braiding of Majorana Fermions in a Josephson Junction Array*, *New J. Phys.* **14**, 035019 (2012).
- [40] T. Hyart, B. van Heck, I. C. Fulga, M. Burrello, A. R. Akhmerov, and C. W. J. Beenakker, *Flux-Controlled Quantum Computation with Majorana Fermions*, *Phys. Rev. B* **88**, 035121 (2013).
- [41] D. J. van Woerkom, A. Geresdi, and L. P. Kouwenhoven, *One Minute Parity Lifetime of a NbTiN Cooper-Pair Transistor*, *Nat. Phys.* **11**, 547 (2015).
- [42] A. P. Higginbotham, S. M. Albrecht, G. Kiršanskas, W. Chang, F. Kuemmeth, P. Krogstrup, T. S. Jespersen, J. Nygård, K. Flensberg, and C. M. Marcus, *Parity Lifetime*

- of Bound States in a Proximitized Semiconductor Nanowire, *Nat. Phys.* **11**, 1017 (2015).
- [43] S. M. Albrecht, E. B. Hansen, A. P. Higginbotham, F. Kuemmeth, T. S. Jespersen, J. Nygård, P. Krogstrup, J. Danon, K. Flensberg, and C. M. Marcus, *Transport Signatures of Quasiparticle Poisoning in a Majorana Island*, *Phys. Rev. Lett.* **118**, 137701 (2017).
- [44] P. O. Boykin, T. Mor, M. Pulver, V. Roychowdhury, and F. Vatan, *New Universal and Fault-Tolerant Quantum Basis*, *Inf. Proc. Lett.* **75**, 101 (2000).
- [45] J. Alicea, Y. Oreg, G. Refael, F. von Oppen, and M. P. A. Fisher, *Non-Abelian Statistics and Topological Quantum Information Processing in 1D Wire Networks*, *Nat. Phys.* **7**, 412 (2011).
- [46] O. Zilberberg, B. Braunecker, and D. Loss, *Controlled-NOT Gate for Multiparticle Qubits and Topological Quantum Computation Based on Parity Measurements*, *Phys. Rev. A* **77**, 012327 (2008).
- [47] J. Eisert, K. Jacobs, P. Papadopoulos, and M. B. Plenio, *Optimal Local Implementation of Non-local Quantum Gates*, *Phys. Rev. A* **62**, 052317 (2000).
- [48] A. Paler, S. Devitt, K. Nemoto, and I. Polian, *Software-Based Pauli Tracking in Fault-Tolerant Quantum Circuits*, in *Design, Automation and Test In Europe Conference and Exhibition (IEEE, New York, 2014)*, pp. 1–4.
- [49] D. Gottesman, *The Heisenberg Representation of Quantum Computers*, Proc. XXII Int. Coll. Group. Th. Meth. Phys. **1**, 32 (1999).
- [50] 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).
- [51] D. Gottesman, Ph.D. thesis, California Institute of Technology, 1997.
- [52] 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).
- [53] G. Duclos-Cianci and D. Poulin, *Fast Decoders for Topological Quantum Codes*, *Phys. Rev. Lett.* **104**, 050504 (2010).
- [54] A. Hutter, J. R. Wootton, and D. Loss, *An Efficient Markov Chain Monte Carlo Algorithm for the Surface Code*, *Phys. Rev. A* **89**, 022326 (2014).
- [55] M. Herold, E. T. Campbell, J. Eisert, and M. J. Kastoryano, *Cellular-Automaton Decoders for Topological Quantum Memories*, *npj Quant. Inf.* **1**, 15010 (2015).
- [56] P. Sarvepalli and R. Raussendorf, *Efficient Decoding of Topological Color Codes*, *Phys. Rev. A* **85**, 022317 (2012).
- [57] J. Marks, T. Jochym-O'Connor, and V. Gheorghiu, *Comparison of Fault-Tolerant Thresholds for Planar Qudit Geometries*, [arXiv:1701.0233](https://arxiv.org/abs/1701.0233).
- [58] 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*, *Quantum Inf. Comput.* **10**, 780 (2010).
- [59] A. Kubica, B. Yoshida, and F. Pastawski, *Unfolding the Color Code*, *New J. Phys.* **17**, 083026 (2015).
- [60] E. T. Campbell, B. M. Terhal, and C. Vuillot, *The Steep Road Towards Robust and Universal Quantum Computation*, [arXiv:1612.07330](https://arxiv.org/abs/1612.07330).
- [61] R. S. Andrist, H. G. Katzgraber, H. Bombin, and M. A. Martin-Delgado, *Tricolored Lattice Gauge Theory with Randomness: Fault Tolerance in Topological Color Codes*, *New J. Phys.* **13**, 083006 (2011).
- [62] R. S. Andrist, H. G. Katzgraber, H. Bombin, and M. A. Martin-Delgado, *Error Tolerance of Topological Codes with Independent Bit-Flip and Measurement Errors*, *Phys. Rev. A* **94**, 012318 (2016).
- [63] A. Steane, *Multiple-Particle Interference, and Quantum Error Correction*, *Proc. R. Soc. Edinburgh, Sect. A* **452**, 2551 (1996).
- [64] A. R. Calderbank and P. W. Shor, *Good Quantum Error-Correcting Codes Exist*, *Phys. Rev. A* **54**, 1098 (1996).
- [65] A. Kubica and M. E. Beverland, *Universal Transversal Gates with Color Codes—A Simplified Approach*, *Phys. Rev. A* **91**, 032330 (2015).
- [66] F. Pastawski and B. Yoshida, *Fault-Tolerant Logical Gates in Quantum Error-Correcting Codes*, *Phys. Rev. A* **91**, 012305 (2015).
- [67] A. M. Meier, B. Eastin, and E. Knill, *Magic-State Distillation with the Four-Qubit Code*, *Quantum Inf. Comput.* **13**, 195 (2013).
- [68] S. Bravyi and J. Haah, *Magic-State Distillation with Low Overhead*, *Phys. Rev. A* **86**, 052329 (2012).
- [69] A. G. Fowler, S. J. Devitt, and C. Jones, *Surface Code Implementation of Block Code State Distillation*, *Sci. Rep.* **3**, 1939 (2013).
- [70] S. Plugge, A. Rasmussen, R. Egger, and K. Flensberg, *Majorana Box Qubits*, *New J. Phys.* **19**, 012001 (2017).
- [71] T. Karzig, C. Knapp, R. M. Lutchyn, P. Bonderson, M. B. Hastings, C. Nayak, J. Alicea, K. Flensberg, S. Plugge, Y. Oreg, C. M. Marcus, and M. H. Freedman, *Scalable Designs for Quasiparticle-Poisoning-Protected Topological Quantum Computation with Majorana Zero Modes*, *Phys. Rev. B* **95**, 235305 (2017).
- [72] P. Bonderson, M. Freedman, and C. Nayak, *Measurement-Only Topological Quantum Computation*, *Phys. Rev. Lett.* **101**, 010501 (2008).
- [73] S. Vijay and L. Fu, *Teleportation-Based Quantum Information Processing with Majorana Zero Modes*, *Phys. Rev. B* **94**, 235446 (2016).
- [74] A. P. Mackenzie and Y. Maeno, *The Superconductivity of Sr₂RuO₄ and the Physics of Spin-Triplet Pairing*, *Rev. Mod. Phys.* **75**, 657 (2003).
- [75] J. Shabani, M. Kjaergaard, H. J. Suominen, Y. Kim, F. Nichele, K. Pakrouski, T. Stankevic, R. M. Lutchyn, P. Krogstrup, R. Feidenhans'l et al., *Two-Dimensional Epitaxial Superconductor-Semiconductor Heterostructures: A Platform for Topological Superconducting Networks*, *Phys. Rev. B* **93**, 155402 (2016).
- [76] F. Amet, C. T. Ke, I. V. Borzenets, J. Wang, K. Watanabe, T. Taniguchi, R. S. Deacon, M. Yamamoto, Y. Bomze, S. Tarucha, and G. Finkelstein, *Supercurrent in the Quantum Hall Regime*, *Science* **352**, 966 (2016).
- [77] G.-H. Lee, K.-F. Huang, D. K. Efetov, D. S. Wei, S. Hart, T. Taniguchi, K. Watanabe, A. Yacoby, and P. Kim, *Inducing Superconducting Correlation in Quantum Hall Edge States*, [arXiv:1609.08104](https://arxiv.org/abs/1609.08104).
- [78] C.-Z. Chang, J. Zhang, X. Feng, J. Shen, Z. Zhang, M. Guo, K. Li, Y. Ou, P. Wei, L.-L. Wang et al., *Experimental Observation of the Quantum Anomalous Hall Effect in a Magnetic Topological Insulator*, *Science* **340**, 167 (2013).

- [79] C.-X. Liu, S.-C. Zhang, and X.-L. Qi, *The Quantum Anomalous Hall Effect: Theory and Experiment*, *Annu. Rev. Condens. Matter Phys.* **7**, 301 (2016).
- [80] L. Fu, *Electron Teleportation via Majorana Bound States in a Mesoscopic Superconductor*, *Phys. Rev. Lett.* **104**, 056402 (2010).
- [81] B. Béri and N. R. Cooper, *Topological Kondo Effect with Majorana Fermions*, *Phys. Rev. Lett.* **109**, 156803 (2012).
- [82] A. Altland, B. Béri, R. Egger, and A. M. Tsvelik, *Multi-channel Kondo Impurity Dynamics in a Majorana Device*, *Phys. Rev. Lett.* **113**, 076401 (2014).
- [83] F. Persson, C. M. Wilson, M. Sandberg, and P. Delsing, *Fast Readout of a Single Cooper-Pair Box Using Its Quantum Capacitance*, *Phys. Rev. B* **82**, 134533 (2010).
- [84] J. Stehlik, Y.-Y. Liu, C. M. Quintana, C. Eichler, T. R. Hartke, and J. R. Petta, *Fast Charge Sensing of a Cavity-Coupled Double Quantum Dot Using a Josephson Parametric Amplifier*, *Phys. Rev. Applied* **4**, 014018 (2015).
- [85] K. D. Petersson, C. G. Smith, D. Anderson, P. Atkinson, G. A. C. Jones, and D. A. Ritchie, *Charge and Spin State Readout of a Double Quantum Dot Coupled to a Resonator*, *Nano Lett.* **10**, 2789 (2010).
- [86] J. I. Colless, A. C. Mahoney, J. M. Hornibrook, A. C. Doherty, H. Lu, A. C. Gossard, and D. J. Reilly, *Dispersive Readout of a Few-Electron Double Quantum Dot with Fast RF Gate Sensors*, *Phys. Rev. Lett.* **110**, 046805 (2013).
- [87] J. H. Béjanin, T. G. McConkey, J. R. Rinehart, C. T. Earnest, C. R. H. McRae, D. Shiri, J. D. Bateman, Y. Rohanizadegan, B. Penava, P. Breul *et al.*, *Three-Dimensional Wiring for Extensible Quantum Computing: The Quantum Socket*, *Phys. Rev. Applied* **6**, 044010 (2016).
- [88] Q. Liu, M. Li, K. Dai, K. Zhang, G. Xue, X. Tan, H. Yu, and Y. Yu, *Extensible 3D Architecture for Superconducting Quantum Computing*, *Appl. Phys. Lett.* **110**, 232602 (2017).
- [89] D. Rosenberg, D. Kim, R. Das, D. Yost, S. Gustavsson, D. Hover, P. Krantz, A. Melville, L. Racz, G. Samach *et al.*, *3D Integrated Superconducting Qubits*, [arXiv:1706.04116](https://arxiv.org/abs/1706.04116).
- [90] A. Robertson, C. Granade, S. D. Bartlett, and S. T. Flammia, *Tailored Codes for Small Quantum Memories*, [arXiv:1703.08179](https://arxiv.org/abs/1703.08179).
- [91] C. Knapp, M. Zaletel, D. E. Liu, M. Cheng, P. Bonderson, and C. Nayak, *The Nature and Correction of Diabatic Errors in Anyon Braiding*, *Phys. Rev. X* **6**, 041003 (2016).

Conclusion

The description of fault-tolerant quantum computing with topological superconductor networks concludes our journey from full quantum computers to logical qubits and lattice surgery, down to the physical components that make up Majorana-based qubits. Ever since the first experiments [41] demonstrated signatures of Majorana zero modes in topological superconductors in 2012, all experimental attempts to build a Majorana-based qubit have, so far, remained unsuccessful. Similarly, currently available non-topological qubits lack the quality and qubit count to construct an error-corrected surface-code qubit capable of significantly increasing the encoded qubit's lifetime, as qubit numbers are below 100 and physical error rates are not sufficiently far below the error threshold.

The requirements of large-scale quantum computing can be daunting. With useful surface-code-based computations requiring hundreds of thousands of physical qubits at a circuit-level error rate of 10^{-3} , several order-of-magnitude improvements are necessary with respect to gate and measurement fidelity, noise and scalability on all qubit platforms. At the same time, there are plenty of reasons for optimism: Not only has the performance of physical qubits steadily improved through experimental progress, but also theoretical developments have significantly lowered the bar for useful quantum computation, with algorithmic improvements dramatically decreasing the number of operations required for useful computations, and improvements in fault tolerance, such as those developed in this thesis, decreasing the overhead of error correction.

There is plenty of room for optimization to obtain further order-of-magnitude overhead reductions on the level of logical operations, circuits, as well as error-correction operations on physical qubits. Since quantum computing with topological codes consists of the distillation of resource states and their consumption, more efficient distillation protocols can reduce the space overhead, as well as the length of computations. Moreover, the distillation of resource states for non-Clifford gates other than T gates can potentially reduce the cost of these gates by many orders of magnitude, as, for instance, the synthesis of a single small-angle rotation by hundreds of T gates [52] can be replaced by the consumption of a few resource states. While the distillation of these states was considered to be very costly [53, 54], recent developments [55, 56] have drastically decreased these costs. Still, the incorporation of small-angle-rotation resource states in the context of a large-scale quantum computer and an estimate of the overhead reduction for computations that rely on small-angle rotations, such as randomly compiled Trotter steps [57] in quantum simulation, is still missing. Furthermore, the implementation and optimization of synthillation protocols [58, 59] that generate resource states capable of performing entire blocks of non-Clifford gates remain largely unexplored.

Apart from algorithmic improvements, the reduction of the overall gate count through circuit optimization can have far-reaching effects, not only reducing the total time cost of a quantum computation, but also, as a consequence, reducing the required code distance and distillation effort, further driving down the space cost as well as the time cost. There exist approaches to

reduce the gate count of a circuit that go beyond the commutation-based schemes discussed in Chapter 1, such as approaches based on phase polynomials [58, 60] or random compiling [57, 61], but these are still subject to optimization, and need to be benchmarked with respect to their application in concrete quantum computations.

The generation of resource states can also be improved through protocols on the level of physical qubits and stabilizer measurements, as the first step of every distillation scheme is the initialization of faulty magic states through state injection. Only few such protocols [21, 24, 25] have been considered. Detecting errors during state injection through post-selection [25] can lower the error rate of undistilled magic states, thereby decreasing the distillation cost to obtain sufficiently good magic states. Moreover, if the noise properties of a specific physical architecture are known, encoding schemes and decoders tailored towards a specific noise model [62, 63] can lower the logical error rate, thereby lowering the required code distance. Finally, finding a suitable replacement for the surface code would help avoid its substantial qubit overhead. Alternative schemes like color codes exist, but, so far, their disadvantages (e.g., higher-weight stabilizers and lack of efficient decoders) tend to prevent them from outperforming surface codes in realistic physical implementations.

The 1920s were the first decade of quantum mechanics, featuring the development of the fundamentals of quantum theory such as wave-particle duality, the uncertainty principle and the Copenhagen interpretation. A century later, the 2020s have the potential to become the decade of quantum computing. With hundreds of quantum computing start-ups, companies and research institutions [64] working to advance quantum computers, the field has recently benefitted from multi-billion-dollar investments [65]. I, for one, remain optimistic that a useful, large-scale quantum computer can be constructed in the coming decade, marking the onset of the second quantum revolution.

Acknowledgments

First and foremost, I would like to thank my PhD advisor Felix von Oppen for the opportunity to work in his group, for mentoring me in the intricacies of academic work, for various insightful scientific discussions, and for enabling me to travel around the world to attend numerous conferences and participate in research collaborations. I could not have asked for a better mentor and PhD supervisor.

Having spent almost 8 years at the Freie Universität Berlin, I would also like to thank my Bachelor's thesis supervisor Jens Eisert and my Master's thesis supervisor Piet Brouwer, who have continued to support me with scientific discussions and collaborations throughout my PhD.

For their patience in dealing with my bureaucratic ineptitude, I would like to express my gratitude to our administrators Gabriele Herrmann, Brigitte Odeh, Annette Schumann-Welde and Marietta Wissmann.

I would also like to thank my friends and colleagues at the Dahlem Center for Complex Quantum Systems and other groups for a wonderful and entertaining time at the institute. There are too many names to list, and I would feel terrible leaving anyone out! Thank you to all of you!

Finally, I would like to thank my parents for their unlimited support in all of my endeavours.

Bibliography

- [1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, by Michael A. Nielsen, Isaac L. Chuang, Cambridge, UK: Cambridge University Press, 2010 (2010).
- [2] P. O. Boykin, T. Mor, M. Pulver, V. Roychowdhury, and F. Vatan, *New universal and fault-tolerant quantum basis*, Inf. Proc. Lett. **75**, 101 (2000).
- [3] M. Reiher, N. Wiebe, K. M. Svore, D. Wecker, and M. Troyer, *Elucidating reaction mechanisms on quantum computers*, PNAS **114**, 7555 (2017).
- [4] R. Babbush, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, A. Paler, A. Fowler, and H. Neven, *Encoding electronic spectra in quantum circuits with linear T complexity*, Phys. Rev. X **8**, 041015 (2018).
- [5] T. D. Ladd, B. Jelezko, R. Laflamme, C. Monroe, and J. L. O'Brien, *Quantum computers*, Nature **464**, 45 (2010).
- [6] A. Y. Kitaev, *Unpaired Majorana fermions in quantum wires*, Sov. Phys. Usp. **44**, 131 (2001).
- [7] R. M. Lutchyn, E. P. A. M. Bakkers, L. P. Kouwenhoven, P. Krogstrup, C. M. Marcus, and Y. Oreg, *Majorana zero modes in superconductor-semiconductor heterostructures*, Nat. Rev. Mater. **3**, 52 (2018).
- [8] C. Knapp, T. Karzig, R. M. Lutchyn, and C. Nayak, *Dephasing of Majorana-based qubits*, Phys. Rev. B **97**, 125404 (2018).
- [9] A. Y. Kitaev, *Fault-tolerant quantum computation by anyons*, Ann. Phys. **303**, 2 (2003).
- [10] E. T. Campbell, B. M. Terhal, and C. Vuillot, *Roads towards fault-tolerant universal quantum computation*, Nature **549**, 172 (2017).
- [11] H. Bombin and M. A. Martin-Delgado, *Topological quantum distillation*, Phys. Rev. Lett. **97**, 180501 (2006).
- [12] S. Bravyi and A. Kitaev, *Universal quantum computation with ideal Clifford gates and noisy ancillas*, Phys. Rev. A **71**, 022316 (2005).
- [13] T. Karzig, C. Knapp, R. M. Lutchyn, P. Bonderson, M. B. Hastings, C. Nayak, J. Alicea, K. Flensberg, S. Plugge, Y. Oreg, C. M. Marcus, and M. H. Freedman, *Scalable designs for quasiparticle-poisoning-protected topological quantum computation with Majorana zero modes*, Phys. Rev. B **95**, 235305 (2017).

- [14] D. Aasen, M. Hell, R. V. Mishmash, A. Higginbotham, J. Danon, M. Leijnse, T. S. Jespersen, J. A. Folk, C. M. Marcus, K. Flensberg, and J. Alicea, *Milestones toward majorana-based quantum computing*, Phys. Rev. X **6**, 031016 (2016).
- [15] 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).
- [16] G. Duclos-Cianci and D. Poulin, *Fast decoders for topological quantum codes*, Phys. Rev. Lett. **104**, 050504 (2010).
- [17] A. Hutter, J. R. Wootton, and D. Loss, *An efficient Markov chain Monte Carlo algorithm for the surface code*, Phys. Rev. A **89**, 022326 (2014).
- [18] M. Herold, E. T. Campbell, J. Eisert, and M. J. Kastoryano, *Cellular-automaton decoders for topological quantum memories*, npj Quant. Inf. **1**, 15010 (2015).
- [19] N. Delfosse and N. H. Nickerson, *Almost-linear time decoding algorithm for topological codes*, arXiv:1709.06218 (2017).
- [20] R. Sweke, M. S. Kesselring, E. P. L. van Nieuwenburg, and J. Eisert, *Reinforcement learning decoder for fault-tolerant quantum computation*, arXiv:1810.07207 (2018).
- [21] C. Horsman, A. G. Fowler, S. Devitt, and R. V. Meter, *Surface code quantum computing by lattice surgery*, New J. Phys. **14**, 123011 (2012).
- [22] C. Vuillot, L. Lao, B. Criger, C. G. Almudever, and B. M. Terhal, *Code deformation and lattice surgery are gauge fixing*, arXiv:1810.10037 (2018).
- [23] H. Bombin, *Topological order with a twist: Ising anyons from an abelian model*, Phys. Rev. Lett. **105**, 030403 (2010).
- [24] A. J. Landahl and C. Ryan-Anderson, *Quantum computing by color-code lattice surgery*, arXiv:1407.5103 (2014).
- [25] Y. Li, *A magic state's fidelity can be superior to the operations that created it*, New J. Phys. **17**, 023037 (2015).
- [26] A. J. Landahl, J. T. Anderson, and P. R. Rice, *Fault-tolerant quantum computing with color codes*, arXiv:1108.5738 (2011).
- [27] X.-L. Qi and S.-C. Zhang, *Topological insulators and superconductors*, Rev. Mod. Phys. **83**, 1057 (2011).
- [28] J. Alicea, *New directions in the pursuit of Majorana fermions in solid state systems*, Rep. Prog. Phys. **75**, 076501 (2012).
- [29] M. Leijnse and K. Flensberg, *Introduction to topological superconductivity and Majorana fermions*, Sem. Sc. Tech. **27**, 124003 (2012).
- [30] C. W. J. Beenakker, *Search for Majorana fermions in superconductors*, Ann. Rev. Cond. Matt. Phys. **4**, 113 (2013).
- [31] T. D. Stanescu and S. Tewari, *Majorana fermions in semiconductor nanowires: fundamentals, modeling, and experiment*, Journal of Physics: Condensed Matter **25**, 233201 (2013).

- [32] S. R. Elliott and M. Franz, *Colloquium: Majorana fermions in nuclear, particle, and solid-state physics*, Rev. Mod. Phys. **87**, 137 (2015).
- [33] R. Aguado, *Majorana quasiparticles in condensed matter*, Riv. Nuovo Cimento **11**, 523 (2017).
- [34] S. B. Bravyi and A. Y. Kitaev, *Quantum codes on a lattice with boundary*, arXiv:quant-ph/9811052 (1998).
- [35] M. H. Freedman, *P/NP, and the quantum field computer*, Proceedings of the National Academy of Sciences **95**, 98 (1998).
- [36] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, *Topological quantum memory*, Journal of Mathematical Physics **43**, 4452 (2002).
- [37] R. Raussendorf, J. Harrington, and K. Goyal, *A fault-tolerant one-way quantum computer*, Annals of physics **321**, 2242 (2006).
- [38] R. Raussendorf and J. Harrington, *Fault-tolerant quantum computation with high threshold in two dimensions*, Phys. Rev. Lett. **98**, 190504 (2007).
- [39] Y. Oreg, G. Refael, and F. von Oppen, *Helical liquids and Majorana bound states in quantum wires*, Phys. Rev. Lett. **105**, 177002 (2010).
- [40] R. M. Lutchyn, J. D. Sau, and S. Das Sarma, *Majorana fermions and a topological phase transition in semiconductor-superconductor heterostructures*, Phys. Rev. Lett. **105**, 077001 (2010).
- [41] V. Mourik, K. Zuo, S. M. Frolov, S. R. Plissard, E. P. a. M. Bakkers, and L. P. Kouwenhoven, *Signatures of Majorana fermions in in hybrid superconductor-semiconductor nanowire devices*, Science **336**, 1003 (2012).
- [42] J. Alicea, Y. Oreg, G. Refael, F. von Oppen, and M. P. A. Fisher, *Non-Abelian statistics and topological quantum information processing in 1D wire networks*, Nat. Phys. **7**, 412 (2011).
- [43] B. Van Heck, A. Akhmerov, F. Hassler, M. Burrello, and C. Beenakker, *Coulomb-assisted braiding of majorana fermions in a Josephson junction array*, New J. Phys. **14**, 035019 (2012).
- [44] S. Plugge, A. Rasmussen, R. Egger, and K. Flensberg, *Majorana box qubits*, New J. Phys. **19**, 012001 (2017).
- [45] S. Vijay, T. H. Hsieh, and L. Fu, *Majorana Fermion surface code for universal quantum computation*, Phys. Rev. X **5**, 041038 (2015).
- [46] L. A. Landau, S. Plugge, E. Sela, A. Altland, S. M. Albrecht, and R. Egger, *Towards realistic implementations of a majorana surface code*, Phys. Rev. Lett. **116**, 050501 (2016).
- [47] D. Gottesman, *The Heisenberg representation of quantum computers*, Proc. XXII Int. Coll. Group. Th. Meth. Phys. **1**, 32 (1999).
- [48] A. G. Fowler, S. J. Devitt, and C. Jones, *Synthesis of arbitrary quantum circuits to topological assembly: Systematic, online and compact*, Scientific Rep. **7**, 10414 (2017).

- [49] A. Paler, I. Polian, K. Nemoto, and S. J. Devitt, *Fault-tolerant, high-level quantum circuits: form, compilation and description*, Quantum Sci. Technol. **2**, 025003 (2017).
- [50] D. Herr, F. Nori, and S. J. Devitt, *Lattice surgery translation for quantum computation*, New J. Phys. **19**, 013034 (2017).
- [51] L. Lao, B. van Wee, I. Ashraf, J. van Someren, N. Khammassi, K. Bertels, and C. G. Almudever, *Mapping of lattice surgery-based quantum circuits on surface code architectures*, Quantum Sci. Technol. **4**, 015005 (2018).
- [52] N. J. Ross and P. Selinger, *Optimal ancilla-free Clifford+T approximation of z-rotations*, arXiv:1403.2975 (2014).
- [53] G. Duclos-Cianci and K. M. Svore, *Distillation of nonstabilizer states for universal quantum computation*, Phys. Rev. A **88**, 042325 (2013).
- [54] E. T. Campbell and J. O’Gorman, *An efficient magic state approach to small angle rotations*, Quantum Sci. Technol. **1**, 015007 (2016).
- [55] E. T. Campbell and M. Howard, *Magic state parity-checker with pre-distilled components*, Quantum **2**, 56 (2018).
- [56] C. Gidney and A. G. Fowler, *Efficient magic state factories with a catalyzed $|CCZ\rangle$ to $2|T\rangle$ transformation*, arXiv:1812.01238 (2018).
- [57] E. T. Campbell, *A random compiler for fast hamiltonian simulation*, arXiv:1811.08017 (2018).
- [58] E. T. Campbell and M. Howard, *Unified framework for magic state distillation and multiqubit gate synthesis with reduced resource cost*, Phys. Rev. A **95**, 022316 (2017).
- [59] J. O’Gorman and E. T. Campbell, *Quantum computation with realistic magic-state factories*, Phys. Rev. A **95**, 032338 (2017).
- [60] M. Amy and M. Mosca, *T-count optimization and reed-muller codes*, arXiv:1601.07363 (2016).
- [61] E. Campbell, *Shorter gate sequences for quantum computing by mixing unitaries*, Phys. Rev. A **95**, 042306 (2017).
- [62] D. K. Tuckett, S. D. Bartlett, and S. T. Flammia, *Ultrahigh error threshold for surface codes with biased noise*, Phys. Rev. Lett. **120**, 050505 (2018).
- [63] D. K. Tuckett, C. T. Chubb, S. Bravyi, S. D. Bartlett, and S. T. Flammia, *Tailoring surface codes for highly biased noise*, arXiv:1812.08186 (2018).
- [64] Quantum computing report, <https://quantumcomputingreport.com/players/>, accessed: 2019-03-10.
- [65] The next decade in quantum computing—and how to play, <https://www.bcg.com/de-de/publications/2018/next-decade-quantum-computing-how-play.aspx>, accessed: 2019-03-10.