Compilation of a simple chemistry application to quantum error correction primitives

Nick S. Blunt, György P. Gehér, and Alexandra E. Moylett*
Riverlane, St Andrews House, 59 St Andrews Street, Cambridge, CB2 3BZ, United Kingdom

(Received 4 September 2023; revised 16 January 2024; accepted 26 February 2024; published 26 March 2024)

A number of exciting recent results have been seen in the field of quantum error correction. These include initial demonstrations of error correction on current quantum hardware and resource estimates which improve understanding of the requirements to run large-scale quantum algorithms for real-world applications. In this work, we bridge the gap between these two developments by performing careful estimation of the resources required to fault-tolerantly perform quantum phase estimation (QPE) on a minimal chemical example. Specifically, we describe a detailed compilation of the QPE circuit to lattice surgery operations for the rotated surface code, for a hydrogen molecule in a minimal basis set. We describe a number of optimizations at both the algorithmic and error correction levels. We find that implementing even a simple chemistry circuit requires 1000 qubits and 2300 quantum error correction rounds, emphasising the need for improved error correction techniques specifically targeting the early fault-tolerant regime.

DOI: 10.1103/PhysRevResearch.6.013325

I. INTRODUCTION

Quantum error correction (QEC), the study of how many noisy physical qubits are used to represent a smaller number of less noisy logical qubits, has seen significant recent developments in a number of directions. One such success is experimental demonstrations of error correction successfully suppressing errors on a real-world quantum device [1]. Another recent development is in careful resource estimates, which have allowed for more accurate estimates of the resources a quantum computer requires to solve problems of significant interest, from estimating chemical properties [2–6] to factoring RSA integers [7,8]. These developments together have helped define both the current state of our abilities to suppress noise on quantum devices, and where we need to get to in order to solve key industrial problems.

There are some natural next steps following the experimental demonstration of a logical quantum memory. Natural follow-ups include implementing basic logical gates: implementing Pauli gates through transversal operations, non-Pauli Clifford gates through lattice surgery techniques [9–15], and non-Clifford gates initially through error mitigation techniques [16] and later through magic state distillation [17–19]. Eventually, a natural goal will be to demonstrate small-scale quantum algorithms, showing that these logical operations can be used to solve a toy application. Understanding the resources required for such an algorithm is important for knowing the point at which small applications can start being solved on fault-tolerant quantum computers, as well as helping us understand the constant factors in the scaling of large quantum algorithms. A number of algorithms have recently been proposed that are aimed specifically at this regime, referred to as "early fault-tolerant" algorithms [20–24]; it is therefore particularly relevant to assess how challenging even minimal applications will be to perform using fault-tolerant operations.

In this work, we estimate the resources required for implementing a small quantum algorithm on a fault tolerant quantum computer, including detailed consideration of how to perform each required operation using lattice surgery. The application we choose is quantum phase estimation (QPE) applied to finding the ground-state energy of the hydrogen molecule. This application is sufficiently small that related circuits without QEC have already been successfully run on current quantum hardware [25,26]. We investigate optimizations of this algorithm at a variety of levels, including algorithmic [2,27–29], gate decompositions [30], compilation to lattice surgery primitives [9–14] and generation of magic states [19]. Our final resource estimates are presented in Fig. 1, looking at different physical error rates and techniques which trade off time and space resource requirements. It is worth noting that when implemented on the surface code, even this small application requires hundreds of physical qubits and thousands of QEC rounds. This shows the significant prefactor associated with quantum error correction, and suggests that in early fault-tolerance further techniques will be required to yield small-scale algorithmic demonstrations [16].

The rest of this paper is laid out as follows. In Sec. II, we review the algorithms and chemical system to be considered, and present the logical quantum circuit. In Sec. III, we describe how to decompose the logical quantum circuits into operations from the Clifford $+T$ gate set and how to implement these gates on the surface code using lattice surgery primitives. In Sec. IV, we estimate the overhead introduced

*alex.moylett@riverlane.com

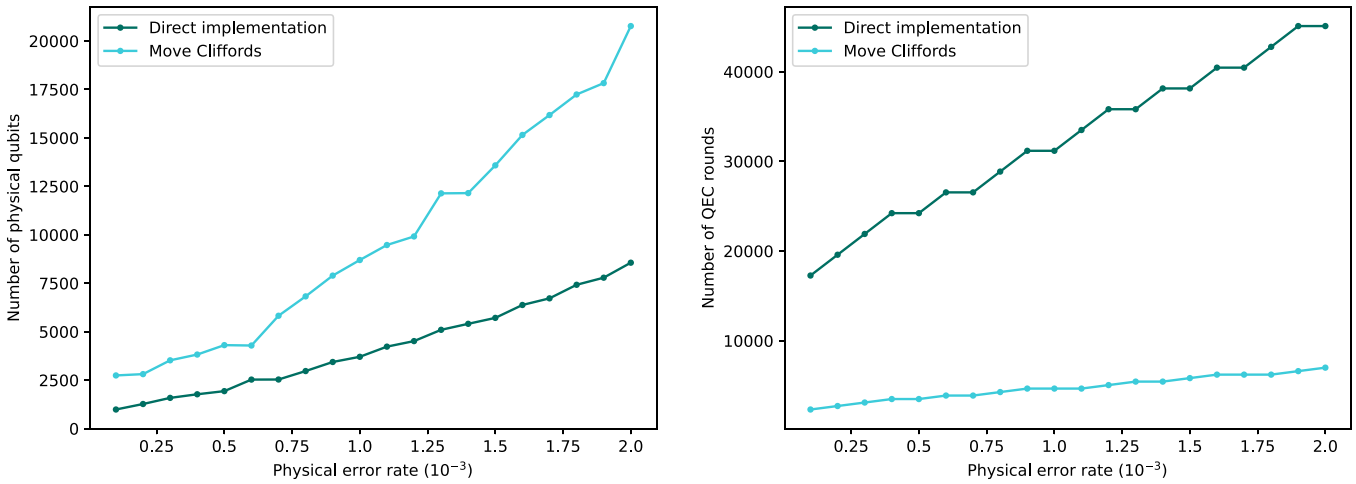


FIG. 1. Estimated cost in physical qubits and time for calculating the ground-state energy of a hydrogen molecule on an error-corrected quantum computer with varying physical error rates, using iterative quantum phase estimation. Methods for implementing logical gates through either directly implementing Clifford and T gates or moving Clifford gates through the circuit are described in Secs. III B and III C, respectively. Further details about estimating these resource requirements are presented in Sec. IV C.

by quantum error correction. Finally, we conclude with some open questions and further directions for research in Sec. V.

II. LOGICAL QUANTUM CIRCUIT

A. Quantum phase estimation

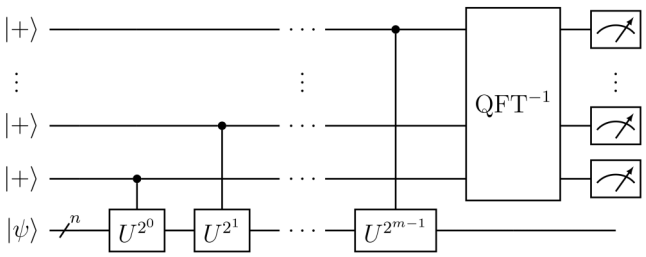
We begin with a brief introduction to the quantum algorithms considered in this study, which are two types of quantum phase estimation (QPE) [31]. QPE is one of the key proposed quantum algorithms for calculating ground and excited-state energies in electronic structure problems. Provided an initial trial state can be prepared that has a sufficiently good overlap with the true ground state (which is usually the case for molecular systems), QPE is capable of obtaining energy estimates to a desired precision in polynomial time with system size. However, the algorithm requires high circuit depths for nontrivial examples, and so has seen less attention compared to variational quantum algorithms in current NISQ applications. For fault-tolerant applications, however, it is often regarded as the algorithm of choice.

We focus on the "textbook" [32] and iterative (semiclassical) QPE algorithms [33–36]. The textbook QPE algorithm is perhaps the best known QPE approach, the circuit for which is presented in Fig. 2(a). The algorithm allows one to measure the eigenphases of some unitary U up to m bits of precision; doing so requires m ancilla qubits, in addition to the n data qubits needed to represent U. At the end of the circuit, an inverse quantum Fourier transform (QFT) is performed and the ancilla qubits are measured. If the input state $|\psi\rangle$ is an exact eigenstate of U, then the measured bits will yield the bits of the corresponding eigenphase. For a nonexact $|\psi\rangle$, the probability of obtaining the desired phase will depend on the overlap between $|\psi\rangle$ and the corresponding exact eigenstate.

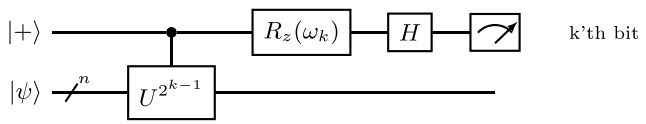
The inverse QFT can also be performed in a semiclassical manner [37]. Using such a semiclassical QFT, the resulting phase estimation algorithm is performed iteratively, obtaining one bit of information about the phase from each iteration. We refer to this approach as iterative quantum phase estimation [36]. Iterative QPE has many of the benefits of the textbook approach, including a Heisenberg-limited running time $\mathcal{O}(\epsilon^{-1})$ for a precision of ϵ, but has the significant benefit that it uses only a single ancilla qubit.

We briefly give some analysis of the iterative QPE approach here. We are interested in estimating the eigenvalues of a Hamiltonian

$$H = \sum_{j=1}^{L} c_j P_j, \tag{1}$$



(a) Textbook QPE circuit.



(b) Iterative QPE circuit.

FIG. 2. QPE circuits used in this paper. In both cases, the state $|\psi\rangle$ is over n qubits. In (b), the circuit is iterated backwards from $k = m$ (in the initial iteration) to $k = 1$ (in the final iteration). The rotation angle in iteration k is $\omega_k = -\pi(0.x_{k+1}x_{k+2}\ldots x_m)$, with $\omega_m = 0$ in the initial iteration. While the ancilla is measured at the end of each iteration, the data qubits remain coherent throughout.

where P_j are n-qubit Pauli operators and c_j are coefficients. We denote the eigenvalues and eigenvectors of H by $\{\lambda_j; |\Psi_j\rangle\}$. We multiply H by a constant t such that $-0.5 \leqslant \lambda_j t \leqslant 0.5$ for all j, which can always be achieved by choosing $1/t = 2\sum_j |c_j|$. We then work with the unitary

$$U = e^{2\pi i H t}. \tag{2}$$

The eigenvalues of U are $e^{2\pi i \phi_j}$, where the range $0 \leqslant \phi_j \leqslant 1$ can be chosen. It is then simple to obtain $\lambda_j t$ from ϕ_j, which only differ due to the wrapping of phases; the normalization of Ht above is chosen to avoid potential ambiguity in this wrapping. Therefore each ϕ_j can be written in binary as

$$\phi_j = 0.\phi_{j1}\phi_{j2}\ldots\phi_{jm}\ldots \tag{3}$$

In iterative QPE, the bits ϕ_{jk} are measured directly using the circuit in Fig. 2(b). The circuit is performed for m iterations in order to obtain m bits of precision for ϕ_j, starting with $k = m$ and iterating backwards to $k = 1$. After each controlled-unitary operation, an $R_z(\omega_k)$ gate is applied to the ancilla with angle

$$\omega_k = -\pi(0.x_{k+1}x_{k+2}\ldots x_m), \tag{4}$$

which depends on the measurement results from previous iterations (and $\omega_m = 0$ in the initial iteration).

The data qubits are prepared in an initial state $|\psi\rangle$, which should be an approximation to the exact state whose energy is to be estimated. We write $|\psi\rangle$ in the eigenbasis of H by

$$|\psi\rangle = \sum_j \nu_j |\Psi_j\rangle. \tag{5}$$

The state of the qubits before the first measurement ($k = m$) is then

$$\frac{1}{2}\sum_j \nu_j[(1 + e^{i2^m\pi\phi_j})|0\rangle + (1 - e^{i2^m\pi\phi_j})|1\rangle] \otimes |\Psi_j\rangle. \tag{6}$$

Consider the simple case where ϕ_j can be represented by exactly m bits, so that $\phi_j = 0.\phi_{j1}\phi_{j2}\ldots\phi_{jm}00\ldots$ In this case $\exp(i2^m\pi\phi_j) = \exp(i\pi\phi_{jm})$ exactly, and the state of the system before measurement is

$$\frac{1}{2}\sum_j \nu_j[(1 + e^{i\pi\phi_{jm}})|0\rangle + (1 - e^{i\pi\phi_{jm}})|1\rangle] \otimes |\Psi_j\rangle. \tag{7}$$

Thus the probabilities of measuring the ancilla as 0 or 1 are

$$P_0 = \sum_j |\nu_j|^2 \cos^2\left(\frac{\pi\phi_{jm}}{2}\right), \tag{8}$$

$$P_1 = \sum_j |\nu_j|^2 \sin^2\left(\frac{\pi\phi_{jm}}{2}\right). \tag{9}$$

Provided that $|\nu_j|$ is sufficiently large for the desired state $|\Psi_j\rangle$, the desired bit will be measured with high probability. The measurement will also project away the contribution from those states $|\Psi_j\rangle$ for which ϕ_{jm} does not match the measured result. It is simple to continue this process for subsequent iterations to $k = 1$. After the final iteration, the probability that all of the bits for the desired ϕ_j were measured is $|\nu_j|^2$. Therefore, for a sufficiently good initial state, and a sufficient number of repetitions, the ground-state energy can be measured with high probability. Further clear analysis is given in [36].

In addition to the textbook and iterative QPE methods, there has been recent progress on statistical phase estimation methods [21,23,38–40]. Compared to the above approaches, such statistical methods allow shorter circuit depth [22,23] and ready combination with error mitigation techniques [26], in exchange for performing many circuits. It has been suggested that these methods are particularly appropriate for early-fault tolerant quantum computers. We do not consider such methods here, but note that they would be interesting to investigate further in the context considered here.

B. Hamiltonian simulation via Trotterization

In this section, we briefly discuss first and second-order Trotterization, and present an optimization to the latter.

We consider n-qubit Hamiltonians of the form of Eq. (1). We specifically denote $H_j = c_j P_j$, so that

$$H = \sum_{j=1}^{L} H_j. \tag{10}$$

In Trotter schemes more generally, each H_j might correspond to a sum of commuting Pauli terms, rather than a single Pauli contribution.

We are concerned with implementing an operator $U = e^{iHt}$, controlled on an ancilla qubit. The well-known first- and second-order Trotter approximations, U_1 and U_2, are

$$U_1 = \prod_{j=1}^{L} e^{iH_j t} \tag{11}$$

and

$$U_2 = \prod_{j=1}^{L} e^{iH_j t/2} \prod_{j=L}^{1} e^{iH_j t/2}, \tag{12}$$

which have errors $\mathcal{O}(t^2)$ and $\mathcal{O}(t^3)$ compared to the exact U, respectively.

Let us consider the number of single-qubit rotations needed to implement the controlled U_1 and U_2 unitaries, as required to perform QPE. Each controlled Pauli rotation, which we shall denote by W_j, can be rewritten as

$$W_j = |0\rangle\langle 0| \otimes \mathbb{1} + |1\rangle\langle 1| \otimes \exp(i\theta_j P_j), \tag{13}$$

$$= \exp(i\theta_j |1\rangle\langle 1| \otimes P_j), \tag{14}$$

$$= \exp(i(\theta_j/2)(\mathbb{1} - Z) \otimes P_j), \tag{15}$$

$$= \exp(i(\theta_j/2)\mathbb{1} \otimes P_j)\exp(-i(\theta_j/2)Z \otimes P_j), \tag{16}$$

which is a product of two multiqubit Pauli rotations. These can be reduced to a single-qubit rotation each after conjugation through an appropriate Clifford [40]. Therefore the cost of each controlled Pauli rotation is 2 single-qubit rotations plus Cliffords, and the number of single-qubit rotations for U_1 is $2L$ per Trotter step.

At first glance it appears that for a given t, the second-order formula requires $4L$ single-qubit rotations to implement. In fact in QPE circuits this is not the case, and the second-order formula can also be implemented with $2L$ rotations, as for the first-order formula, but with better error suppression. This trick was introduced in Ref. [27] and is known as

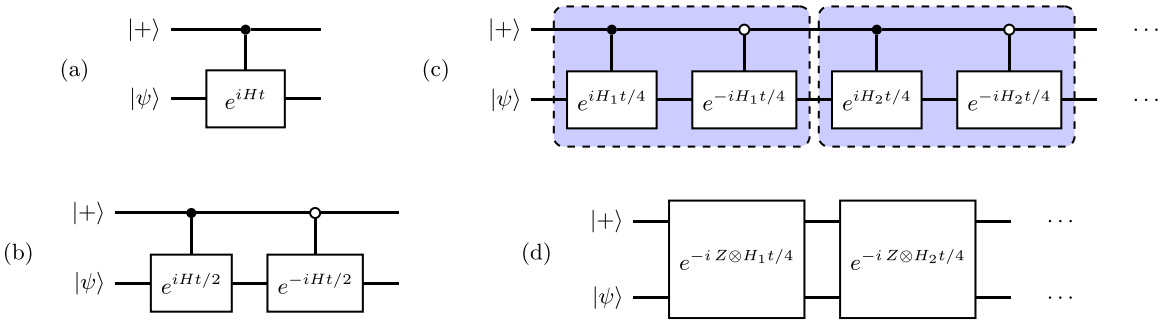


FIG. 3. Circuit diagrams demonstrating reduction of the controlled time evolution operator in QPE with second-order Trotterization. (a) The time evolution operator controlled on an ancilla, and acting on an initial trial state $|\psi\rangle$, which can be equivalently replaced by (b) in phase estimation circuits. For the second-order Trotter formula, this can be further reduced to the circuit (c). Lastly, each pair of boxed terms can be expressed as a single multiqubit Pauli rotation (d).

directionally-controlled phase estimation. It was expanded on in Refs. [2,28] and also used in Ref. [29].

We briefly give a derivation of the directionally controlled approach. The general procedure is presented in Fig. 3. We consider the controlled time evolution operator in Fig. 3(a). The state of the qubits at the end of this circuit is

$$|\Psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle \otimes |\psi\rangle + |1\rangle \otimes e^{iHt}|\psi\rangle). \tag{17}$$

Now, note that we can apply $e^{-iHt/2}$ to the data qubits in Fig. 3(a) without affecting any measurement outcomes; since this operator commutes with all controlled-e^{iHt} gates, it can be moved to the end of the circuit where it has no effect on the measurement of the ancilla. With this additional operator applied, the final state of the qubits is

$$|\Psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle \otimes e^{-iHt/2}|\psi\rangle + |1\rangle \otimes e^{iHt/2}|\psi\rangle), \tag{18}$$

and we see that we can work with circuit in Fig. 3(b) instead.

We next expand $e^{iHt/2}$ via its Trotter formula,

$$e^{iHt/2} \approx V_K \dots V_2 V_1, \tag{19}$$

where K is the number of terms in the Trotter product formula, equal to L for the first-order formula and $2L$ for second-order formula. Then,

$$\begin{aligned}|\Psi\rangle &= |0\rangle \otimes (V_K \dots V_2 V_1)^\dagger |\psi\rangle + |1\rangle \otimes (V_K \dots V_2 V_1)|\psi\rangle \\ &= |0\rangle \otimes V_1^\dagger V_2^\dagger \dots V_K^\dagger |\psi\rangle + |1\rangle \otimes V_K \dots V_2 V_1 |\psi\rangle. \end{aligned} \tag{20}$$

For even-order Trotter formulas the string of operators $V_K \dots V_2 V_1$ is symmetric, so that $V_j = V_{K-j+1}$, and the expansion is unchanged when the order of the terms is reversed. Therefore, for the second-order Trotter formula (but *not* the first-order formula), we can write

$$|\Psi\rangle = |0\rangle \otimes V_K^\dagger \dots V_2^\dagger V_1^\dagger |\psi\rangle + |1\rangle \otimes V_K \dots V_2 V_1 |\psi\rangle, \tag{21}$$

which is equivalent to the circuit in Fig. 3(c). Lastly, note that the paired operators in Fig. 3(c) can each be expressed as

$$e^{i|1\rangle\langle 1|\otimes H_i t/4} e^{-i|0\rangle\langle 0|\otimes H_i t/4} = e^{-iZ\otimes H_i t/4}, \tag{22}$$

which can be reduced to a rotation on a single qubit plus Clifford gates. Therefore, application of the second-order Trotter formula in QPE can be performed with $2L$ rotations, which is equal to the number required for the first-order Trotter formula. In addition, the Trotter expansion is applied to the operator $e^{iHt/2}$ instead of e^{iHt}, resulting in lower Trotter error.

C. The hydrogen molecule

We next define the Hamiltonian that we will consider throughout this paper. As an application of QPE, we will consider the common task of finding the ground-state energy of an electronic structure Hamiltonian. Such a Hamiltonian can be defined in second-quantized form as

$$H = h_0 + \sum_{pq} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{pqrs} h_{pqrs} a_p^\dagger a_q^\dagger a_s a_r, \tag{23}$$

where p, q, r, and s label spin orbitals. The coefficient h_0 defines the nuclear-nuclear contribution (which is just a number due to the Born-Oppenheimer approximation), and h_{pq} and h_{pqrs} are one- and two-body integrals, respectively. The form of these integrals are well known from quantum chemistry [41].

In this paper, we are concerned with compiling a minimal chemistry problem to lattice surgery operations, including visualization of the patch layout. We therefore consider the hydrogen molecule H_2 in a STO-3G basis, which is a prototypical minimal molecular example, consisting of two electrons in two spatial orbitals, or four spin orbitals. We use an equilibrium geometry with an internuclear distance of 0.7414 Å.

The fermionic Hamiltonian in Eq. (23) must be mapped to a qubit Hamiltonian for use in QPE. Because the minimal basis for H_2 consists of four spin orbitals, direct mappings will result in a Hamiltonian with four qubits. However, as shown by Bravyi *et al.* [42], the qubit Hamiltonian for this problem can be reduced to just a single-qubit operator. This can be seen from symmetry arguments; the H_2 Hamiltonian (in this nonrelativistic approximation) commutes with spin and particle-number operators, and also has spatial symmetry. Each of these symmetries allows one qubit to be tapered. More precisely, labeling the bonding and antibonding orbitals as ψ_g and ψ_u, and ordering the spin orbitals as $\psi_{g\uparrow}$, $\psi_{g\downarrow}$, $\psi_{u\uparrow}$, $\psi_{u\downarrow}$ (that is, using a spin-interleaved arrangement), the only determinants that contribute to the ground-state wave function are $|1100\rangle$ and $|0011\rangle$, and these two states can be represented by a single qubit. A more general approach for tapering qubits due to $\mathbb{Z}_2$ symmetries is given in Ref. [42].

The Hamiltonian used takes the form

$$H = c_1 Z + c_2 X, \tag{24}$$

with $c_1 = 0.78796736$ and $c_2 = 0.18128881$, and we have neglected the identity contribution. Note that an identical qubit Hamiltonian was considered in Ref. [25], which performed textbook QPE on a neutral-atom quantum computer.

D. Overall logical circuit

Using the second-order Trotter formula techniques described in Sec. II B, we derive logical circuits for both textbook and iterative quantum phase estimation. We choose a time step $t = \pi/(c_1 + c_2)$, where c_1 and c_2 are defined in Eq. (24), in order to ensure that eigenvalues of Ht are in the range $[-\pi, \pi]$. In this simple application, we take just a single time step in the Trotter expansion of e^{iHt}. We also perform QPE for just three bits of accuracy in the energy. These simplifications will of course lead to large errors in the final energy estimate; indeed, after removing rescaling factors, using three bits of precision means that the energy can only be estimated to precision $(c_1 + c_2)/4 = 0.242$ Ha. Here, we are primarily interested in understanding the required circuits in terms of lattice surgery primitives. Increasing the number of Trotter steps, or bits of precision, does not provide further insight beyond increasing the circuit depth (and number of ancilla qubits, in the case of textbook QPE).

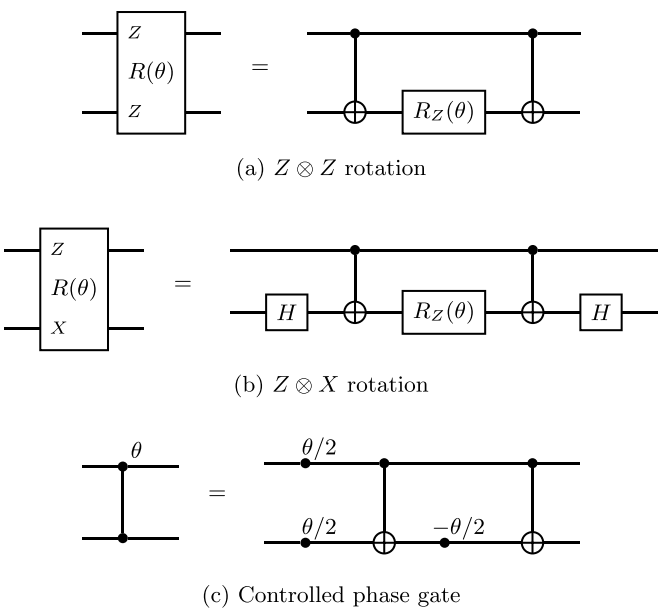
To implement $e^{-ic_1 Z\otimes Z t/4}$ and $e^{-ic_2 Z\otimes X t/2}$ we use rotation operations $R_{Z\otimes Z}$ and $R_{Z\otimes X}$, defining $R_P(\theta) = e^{-iP\theta/2}$. Thus we have rotation angles $\theta_1 = tc_1/2$ and $\theta_2 = tc_2$ for the $Z \otimes Z$ and $Z \otimes X$ rotations, respectively. We also make a minor optimization by combining pairs of $R_{Z\otimes Z}(\theta_1)$ rotations into a single $R_{Z\otimes Z}(2\theta_1)$ rotation where possible. Figures for the logical circuits are provided in Appendix A.

For both iterative and textbook QPE circuits, the gates can be grouped into three types: Pauli gates such as the X gate, non-Pauli Clifford gates such as the Hadamard and $S^\dagger$ gates, and non-Clifford gates such as the two-qubit rotations and $T^\dagger$ gates. These different types of gates require different techniques to be implemented on the surface code, which we shall detail further in Sec. III.

III. IMPLEMENTING LOGICAL GATES

In this section, we discuss how to implement the logical circuits presented in Sec. II D using operations available on the surface code. The surface code represents logical qubits as patches of $d \times d$ data qubits, where d is the distance of the code [43,44]. Stabilizers consist of weight-4 X and Z measurements on the patch, and logical X and Z observables are defined along the horizontal and vertical boundaries of the patch [9]. The surface code has proven to be a popular candidate for fault tolerant quantum algorithms, due to both its high threshold and low connectivity requirements. In particular, a number of resource estimation papers use the surface code as the basis for estimating the overhead from quantum error correction [4–8].

This section proceeds as follows. First, we approximately decompose the logical gates into a sequence of Clifford and T gates. Then we consider two potential methods for implementing these gates: in Sec. III B, we implement the Clifford and T gates directly using native lattice surgery operations; whereas in Sec. III C, we use commutation relations to remove Clifford operations from the circuit, at the cost of needing to implement more general T-like operations.



(a) $Z \otimes Z$ rotation

(b) $Z \otimes X$ rotation

(c) Controlled phase gate

FIG. 4. Decompositions of parameterized (a) $Z \otimes Z$ rotations, (b) $Z \otimes X$ rotations, and (c) controlled-phase gates into Clifford operations and single-qubit Z rotations. Note that while single-qubit Z rotations are equivalent to single-qubit phase gates up to a global phase, the two-qubit $Z \otimes Z$ rotation is different from a controlled phase gate due to local phases.

A. Decomposition to Clifford and T gates

Ideally we would want to implement logical quantum operations transversely on our error-correcting code, applying the operation to each physical qubit(s) in turn. Unfortunately, the Eastin-Knill theorem shows that this is not possible for any quantum error-correcting code [45]. In the case of the surface code, the logical gates which can be implemented transversely are single-qubit Pauli gates if the code distance is odd. Other gates within the Clifford group can be implemented on the surface code via lattice surgery operations such as patch deformation [9], but non-Clifford gates such as the T gate cannot be implemented in an error-corrected fashion.

However, it is possible to approximately decompose an arbitrary unitary operation into a sequence consisting of Clifford gates and the single-qubit T gate. This was shown for arbitrary gates originally using the Solvay-Kitaev theorem [46], and a number of improvements have been subsequently shown for both single- and multiqubit gates [30,47].

In the case of the QPE circuits in Sec. II D, the circuits contain a mixture of Clifford and non-Clifford gates. As Clifford gates can be implemented on the surface code via lattice surgery and patch-deformation techniques, such as the ones that we shall describe in Sec. III B, we only need to decompose the non-Clifford gates. Both the textbook and iterative QPE circuits consist of a series of two-qubit Pauli rotations as

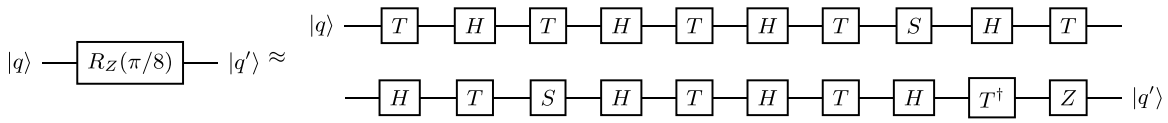


FIG. 5. Example approximate decomposition of a $\pi/8$ Z rotation into a sequence of single-qubit Clifford gates and T gates using the GRIDSYNTH software package with three bits of precision. Global phases have been omitted, and some optimizations have been applied to combine multiple S and T gates.

part of the Trotter expansion. In textbook QPE, there are controlled phase gates after the two-qubit rotations, to implement the inverse quantum Fourier transform. In iterative QPE, the two-qubit rotations are followed by (classically conditioned) single-qubit phase gates to implement a semiclassical version of the inverse Fourier transform.

We decompose the non-Clifford gates in two steps. First, we exactly compile the two-qubit operations into Clifford gates and single-qubit Z rotations and phase gates using circuit identities presented in Fig. 4. Second, we use the GRIDSYNTH software package to approximately decompose the single-qubit Z rotations into sequences of one-qubit Clifford and T gates [30]. Note that the two-qubit $Z \otimes Z$ rotations require a different decomposition to the controlled phase gates, due to differences in local phases. In comparison, single-qubit Z rotations are equivalent to single-qubit phase gates up to a global phase $R_Z(\theta) = e^{-i\theta/2}P(\theta)$, and can therefore be decomposed using the same techniques. An example of using GRIDSYNTH to approximately decompose a single-qubit Z rotation into the Clifford and T gate set is provided in Fig. 5.

There is a trade-off to be made between the accuracy of decompositions generated by GRIDSYNTH and the number of gates required. GRIDSYNTH approximates a single rotation $R_Z(\theta)$ up to error ϵ in the operator norm with typically $3\log_2(1/\epsilon) + O(\ln(\ln(1/\epsilon)))$ non-Clifford gates [30]. To get an understanding of how this extends to a whole circuit, we ran simulations of the textbook and iterative QPE circuits with GRIDSYNTH decompositions of varying accuracy, from 1 bit to 32 bits. For each number of bits of accuracy, we generate 1000 circuits with the single-qubit rotations decomposed to that degree of accuracy, and simulate each circuit 10,000 times.

The results are presented in Fig. 6. In Fig. 6(a), we take the total variation distance between the output distributions of the decomposed circuits with that of the perfect QPE circuit. From this we see that for both textbook and iterative QPE, the total variation distance reduces quickly to approximately 8.3×10^{-3} at 10 bits of precision per gate decomposition, but tails off beyond this value. This is due to finite precision used when estimating the total variation distance from samples. In the following results, we choose 10 bits of precision for the decomposition of phase gates, as it provides sufficient overall total variation distance for purpose of this circuit.

We also present the number of gates required for each gate decomposition accuracy in Fig. 6(b). For 10 bits of precision, there are approximately 1300 and 1000 logical gates for textbook and iterative QPE, respectively. Fewer logical gates can also be used at the cost of increased error; for example, fewer than 1,000 logical gates can be achieved with 5 bits of precision per rotation: 870 gates for textbook QPE, and 740 gates for iterative QPE. The total variation distance at 5 bits of precision is 2.4%.

The results in Fig. 6(b) also show that for this particular circuit iterative QPE requires fewer gates than textbook QPE regardless of decomposition accuracy. This is due to the fact that the inverse QFT step of textbook QPE requires

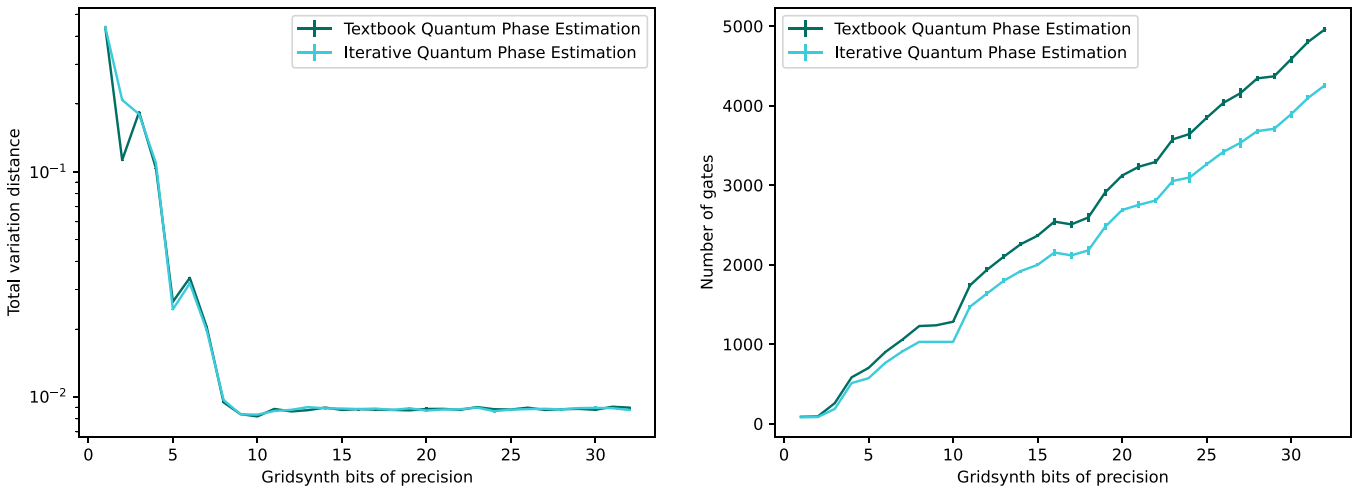


FIG. 6. Performance of GRIDSYNTH on textbook and iterative QPE, for increasing bits of precision in the GRIDSYNTH decomposition (with 3 bits of precision used in each QPE circuit). (a) Comparison of the decomposed circuits to the exact circuits in terms of total variation distance of the output distributions. (b) The number of gates in the overall circuit.

two-qubit controlled phase rotations around fixed angles θ. These are subsequently decomposed into smaller rotations $\theta/2$ and $-\theta/2$, which are then approximately decomposed using GRIDSYNTH. In comparison, iterative QPE works with single qubit phase rotations θ which are classically controlled. As these single angles are larger than those used for the single-qubit rotations in textbook QPE, fewer gates are required to decompose them up to a desired accuracy. For this particular QPE circuit, which is only performed to three bits of accuracy, the smallest rotation angle required beyond the Hamiltonian simulation step in the iterative QPE circuit is $-\pi/4$, which can be implemented as a single $T^\dagger$ gate. Hence for the rest of this paper we shall primarily focus on iterative QPE.

Finally, for anyone curious to see an example of the complete logical circuit, we have included example QASM circuits in the supplementary material to this paper [48], including an iterative QPE circuit with phase gates decomposed up to 10 bits of precision. This circuit features a total of 1029 operations, of which there are 13 X gates, 169 Z gates, 34 CNOT gates, 411 Hadamard gates, 13 $S/S^\dagger$ gates, 386 $T/T^\dagger$ gates, and three measurements in the Z basis. This is the circuit we will estimate the resources for in Sec. IV. Note that GRIDSYNTH is a randomized process, and so different runs might produce different gate decompositions than presented here.

B. Directly implementing Clifford and T gates

Next, we consider methods to implement Clifford and T gates in the logical circuit. These can either be applied directly, or can be moved to the end of the logical circuit [12]. In this section, we first discuss the time and space cost of directly implementing both Clifford and T gates. Both of these estimates will be calculated in terms of the code distance d. The approach of moving Clifford operations will then be considered in Sec. III C.

The simplest gates to implement on the surface code are single-qubit Pauli gates. These operations can be implemented by either applying the corresponding Pauli gate to all data qubits if the distance d is odd, or, if the distance d is even, by tracking their values in software. Due to their simplicity, we shall not focus on how to implement them in this section. Likewise, preparation and measurement of a logical qubit in either the Z or X basis can be done in a single QEC round by preparing or measuring all data qubits in that basis. In many cases these operations can even be implemented at a cost of no additional QEC rounds, by preparing the data qubits at the start of the following round or measuring the data qubits at the end of the preceding round. Note however that preparing or measuring a logical qubit in the Y basis is more complicated [14].

It is important to note that not every Clifford gate presented in Sec. III will be directly implemented. Any sequence consisting of only Z, $S/S^\dagger$, and $T/T^\dagger$ gates can be implemented at the cost of implementing a single T gate (as shown in Appendix B). Thus we can think of the sequences of gates generated by GRIDSYNTH such as those shown in Fig. 5 as equivalent to sequences of alternating Hadamards and T-like gates.

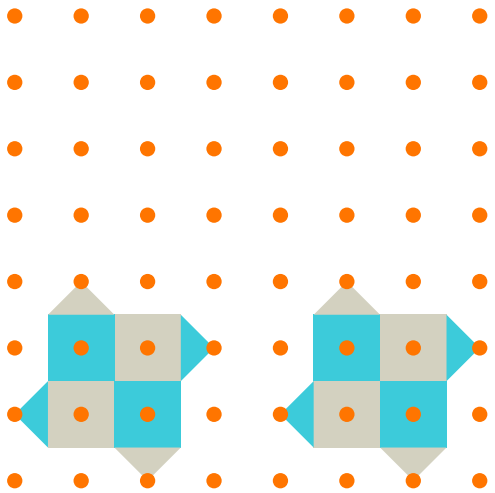


FIG. 7. Layout of logical qubits as distance $d = 3$ surface code patches which can be used when directly implementing Clifford and T operations. Orange dots represent qubits used for measuring stabilizers, which are represented by squares and triangles. X and Z stabilizers are coloured in grey and blue, respectively. Data qubits are not shown, but lie on the corners of the stabilizers. Additional qubits lie outside this space for generating states required for T gates, as detailed in Sec. IV A.

Before discussing how to implement non-Pauli gates, we present how our logical qubits are arranged on a quantum processor with nearest-neighbor connectivity. For iterative QPE, we have two logical qubits, each of which is represented by a $d \times d$ patch. The primary lattice surgery operations we utilize are for implementing joint $Z \otimes Z$ measurements. We arrange our logical qubits as $d \times d$ patches such that performing joint measurements with the horizontal observable is easy. We also introduce two additional spaces of $d \times d$ data qubits, which can be used as both routing space for performing joint measurements with the vertical operator, and for additional qubits required for implementing logical gates. We have the layout in Fig. 7, which for distance d uses a total of $(2d + 2)^2$ data qubits, or $2(2d + 2)^2$ physical qubits including those used for measurement.

1. CNOT gate

A CNOT gate between a control qubit c and target qubit t can be implemented based on two-qubit joint Pauli measurements [9,11], see Fig. 8. Namely, an auxiliary qubit a is initialized in the $|+\rangle$ state, followed by two joint measurements: $Z_c \otimes Z_a$ and $X_t \otimes X_a$. Finally, the auxiliary qubit is measured out in the Z basis, and Pauli corrections are applied based on the outcomes.

This operation can be implemented on our patches via the protocol shown in Fig. 9. In Fig. 9(b), we use the routing space to initialize an additional patch in the $|+\rangle$ state. We then use a merge-and-split operation between the horizontal boundaries of the control patch and the auxiliary patch to perform the $Z \otimes Z$ measurement, and at the same time grow and shrink the target patch to move it into the routing space. Next, we use another merge-and-split operation between the vertical boundaries of the target patch and the auxiliary

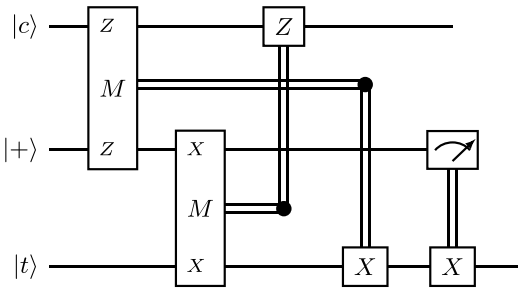
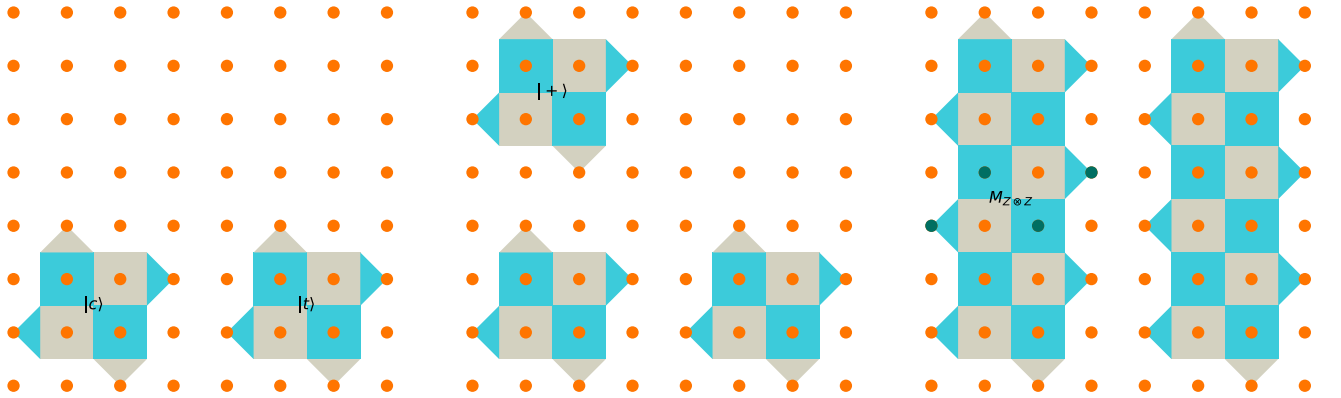


FIG. 8. Circuit for implementing a CNOT gate through joint $Z \otimes Z$ and $X \otimes X$ measurements. An additional qubit initialized in the $|+\rangle$ state is required.

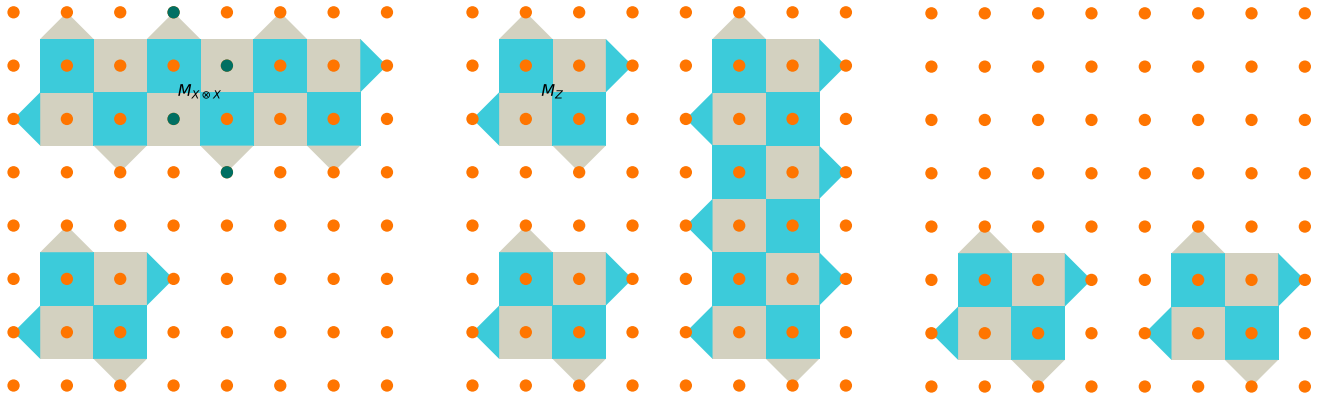
patch to implement the $X \otimes X$ measurement. Finally, we measure out the auxiliary patch and at the same time use patch growing and shrinking to move the target qubit back to its original space. The remaining Pauli operations can either be applied transversely at the start of the next operation if the distance d is odd, or simply tracked in software if the patch distance d is even. The operations for growing and joining patches require d QEC rounds in order to protect the code from both qubit and measurement errors, the operations for splitting and shrinking patches as well as the single-qubit logical X measurement each require a single QEC round, and the Pauli operations at the end of the circuit are effectively free, meaning a total of $3d + 4$ QEC rounds are required to implement the CNOT gate.

2. *Hadamard gate*

The Hadamard gate is a Clifford gate whose role is to swap the X and Z observables of a qubit. Naïvely, this can be achieved on a surface code patch by applying a Hadamard operation transversely to all data qubits on the patch, as shown in Fig. 10(a). However, this has the side-effect of swapping the X and Z stabilizers as well as the logical observables, resulting in a different patch to the one we started with and making joint patch operations such as those used for the CNOT in Sec. III B 1 more complicated. This effect of swapping the stabilizers can be seen by comparing the patches in Figs. 10(a) and 10(b).

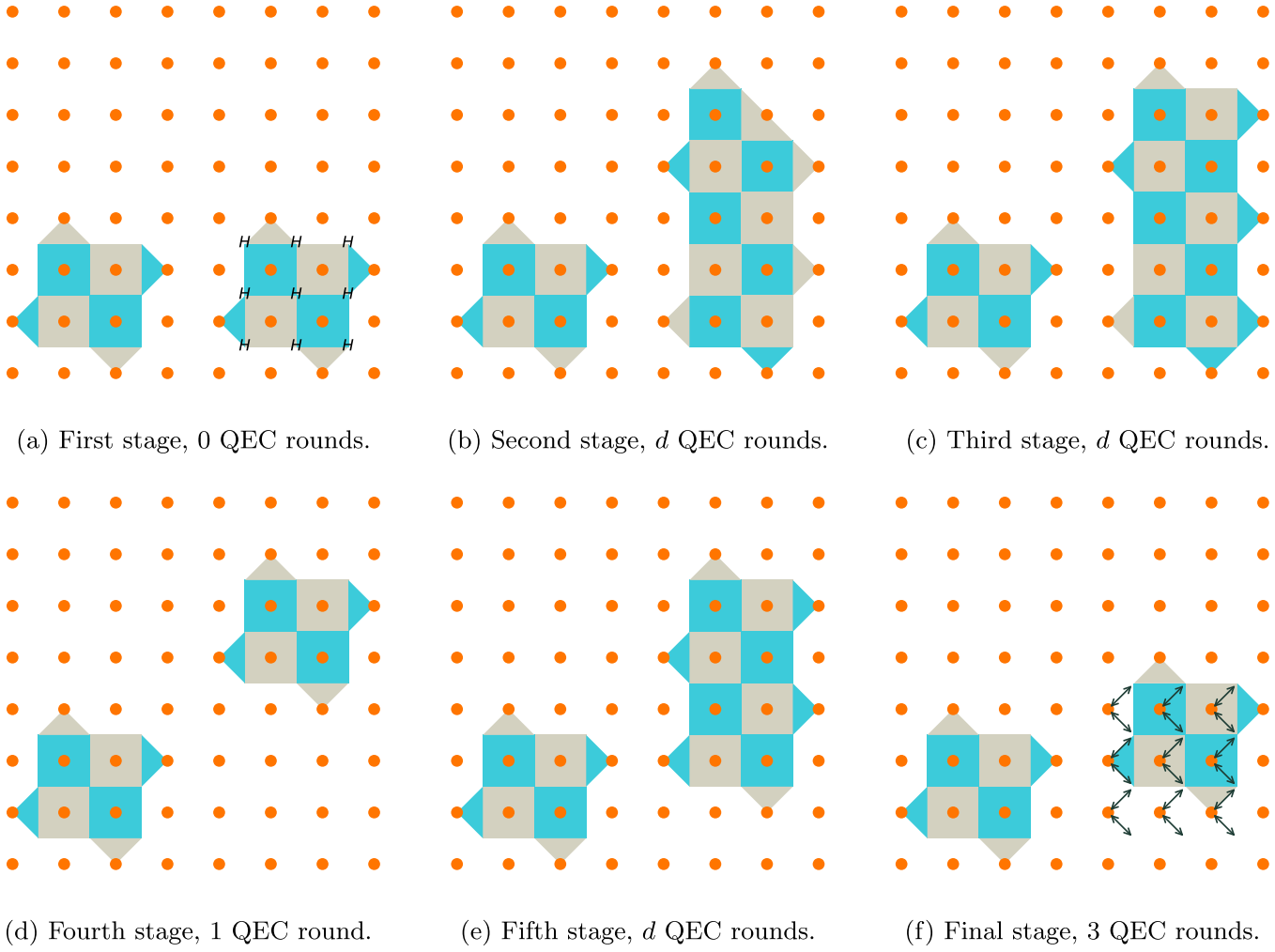


(a) Initial arrangement. (b) First stage, one QEC round. (c) Second stage, $d + 1$ QEC rounds.



(d) Third stage, $d + 1$ QEC rounds. (e) Final stage, d QEC rounds. (f) Final arrangement, one QEC round.

FIG. 9. Implementation of a CNOT via lattice surgery operations, with the left patch of (a) being the control qubit and the right patch being the target qubit. Green dots represent stabilizer measurements whose outcomes produce the result of the joint logical Pauli measurement.



(a) First stage, 0 QEC rounds. (b) Second stage, d QEC rounds. (c) Third stage, d QEC rounds.

(d) Fourth stage, 1 QEC round. (e) Fifth stage, d QEC rounds. (f) Final stage, 3 QEC rounds.

FIG. 10. Implementation of a Hadamard gate on the right logical qubit via a transversal Hadamard gate, a series of patch deformations, and two transversal SWAP gates. Arrows denote SWAP gates between pairs of neighboring qubits. In (f), the first QEC round is to shrink the patch, and the subsequent two QEC rounds occur after each round of SWAP gates.

If we rotated the patch by 90° around the central data qubit after applying the transversal Hadamard gates, then we would have implemented the logical Hadamard gate. However, this is not possible on a physical device. Instead, we use a patch deformation technique, which we present in Figs. 10(b)–10(f), to achieve the same effect [49]. First in Fig. 10(b), we grow the patch into a longer one with length $2d + 1$. At the same time we move the corner at the top right in the original patch to the top left in the longer patch. Next in Fig. 10(c), we use patch deformation to move the corner on the bottom-right up to the top-right. At this stage the logical observables have changed directions from vertical to horizontal and vice versa. Next, we shrink the patch down in Fig. 10(d). Now we have the X and Z logical observables swapped, with the stabilizers in their original positions, but the whole patch has been shifted upwards.

To move this patch back to its original position, we start by growing and shrinking the patch in Figs. 10(e) and 10(f), but this leaves the patch one row of stabilizers higher than it originally was. To correct this, we use two rounds of SWAP gates to swap the data qubits with neighboring measurement qubits, as shown in Fig. 10(f).

The most expensive parts of this process are the stages that involve patch growing and corner movement, which require d QEC rounds each. Since in general two-qubit gates are much noisier than one-qubit gates, the transversal Hadamard at the start of this sequence does not require any QEC rounds. Finally, patch shrinking and transversal SWAP gates each require a single QEC round, thus requiring a total of $3d + 4$ QEC rounds.

3. $S/S^\dagger$ gate

The S gate, also known as the $\sqrt{Z}$ gate, is a Clifford gate that applies a phase of i to the $|1\rangle$ state. Like the Hadamard gate, this gate also cannot be implemented transversely on the surface code.

There are various ways of implementing the S gate using patch deformation, similarly to implementing the Hadamard in Sec. III B 2 [14,49]. However, these require extending the X observable of a patch, and therefore require moving the patch into the routing space and back. Instead, we consider a different technique, which uses an additional patch in the $|Y_+\rangle = (|0\rangle + i|1\rangle)/\sqrt{2}$ state [12]. We then perform a joint

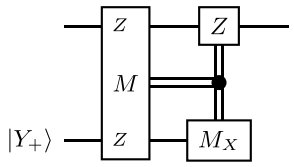


FIG. 11. Circuit for implementing an S gate through a joint $Z \otimes Z$ measurement. An additional qubit initialized in the $|Y_+\rangle = (|0\rangle + i|1\rangle)/\sqrt{2}$ state is required. Note that the $S^\dagger$ gate can be implemented by inverting the condition under which the Z gate is applied.

$Z \otimes Z$ measurement between this qubit and our qubit, and measure this auxiliary qubit in the X basis. Finally, we apply a Z correction depending on the outcomes of the two measurements. A circuit describing this operation is presented in Fig. 11.

Note that unlike Z and X basis states, the $|Y_+\rangle$ state cannot be generated in a single QEC round. Instead, we utilize a different technique to generate Y basis states in $d/2+2$ rounds with no additional qubits [14]. With this additional patch, we can implement the logical S gate using the process described in Fig. 12. Generating the $|Y_+\rangle$ state in Fig. 12(a) takes $d/2+2$ QEC rounds, the joint measurement in Fig. 12(b) takes d QEC rounds, and measuring the $|Y_+\rangle$ state in the X basis takes a single QEC round. Any Z correction can be applied in software at no additional cost, so the total number of QEC rounds required is $3d/2+3$. Finally, note that the $S^\dagger = SZ$ gate can also be implemented at no additional cost, by simply inverting the conditions under which the Z correction is applied.

4. $T/T^\dagger$ gate

The T gate is a nontrivial gate to implement on the surface code as it cannot be performed either transversely or via lattice surgery operations such as patch deformation. Instead, we introduce an auxiliary qubit initialized in the $|T\rangle = (|0\rangle + e^{i\pi/2}|1\rangle)/\sqrt{2}$ state. With this $|T\rangle$ state prepared, we can implement the T gate using techniques similar to those for implementing the S gate in Sec. III B 3. The circuit is presented

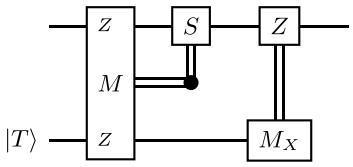
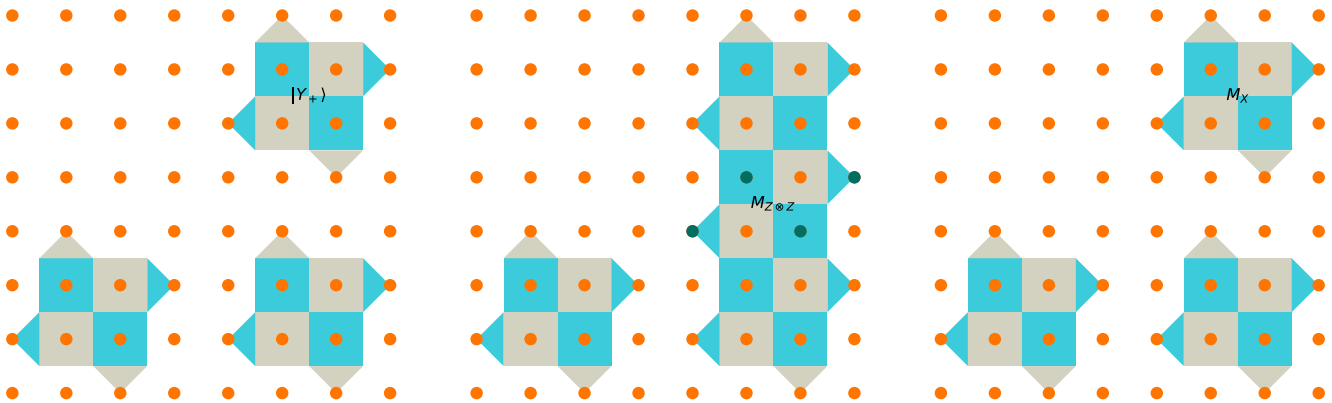


FIG. 13. Circuit for implementing a T gate through a joint $Z \otimes Z$ measurement. An additional qubit initialized in the $|T\rangle = (|0\rangle + e^{i\pi/4}|1\rangle)/\sqrt{2}$ state is required.

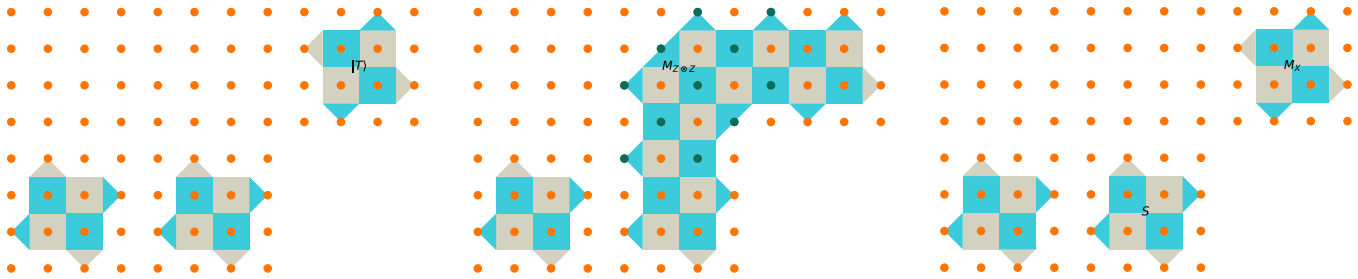
in Fig. 13 [12]. First, we perform a joint $Z \otimes Z$ measurement between the data qubit and the auxiliary qubit. Next, we perform an S gate conditioned on the result of this measurement outcome. Finally, we measure out the auxiliary qubit in the X basis, and depending on this measurement outcome apply a final Z gate to the data qubit. However, while the $|Y_+\rangle$ state can be prepared on the surface code in a fault-tolerant way in $d/2+2$ QEC rounds, the $|T\rangle$ state cannot be prepared on the surface code in an error-corrected fashion, and thus additional work is required in order to prepare a high-quality $|T\rangle$ state. We shall detail this further in Sec. IV A.

We can implement this circuit on our patch layout using the process shown in Fig. 14. Note that the patch for the $|T\rangle$ state is not stored in the routing space like the $|Y_+\rangle$ is in Fig. 12. This is because unlike the $|Y_+\rangle$ state, the $|T\rangle$ state cannot be generated in a fault tolerant process, and instead needs to be generated elsewhere and stored outside of the routing space until it is required. Also note that the patch for the $|T\rangle$ state in Fig. 14(a) is rotated compared to the patches for our data qubits, such that the vertical observable on the auxiliary patch matches the horizontal observable on our data patches. We use this to perform a joint $Z \otimes Z$ measurement between our auxiliary patch and our data patch via merge-and-split operations in Fig. 14(b). Finally, in Fig. 14(c) we measure our auxiliary patch in the X basis, and at the same time we potentially apply an S correction using the methods described in Sec. III B 3. As with the CNOT gate presented in Sec. III B 1, the Z operation is effectively free as it can be either tracked in software or implemented transversely. The joint measurement requires d QEC rounds, the X measurement requires a single QEC



(a) First stage, $d/2+2$ QEC rounds. (b) Second stage, d QEC rounds. (c) Third stage, one QEC round.

FIG. 12. Implementing an S gate on a logical patch. In (a), a patch in an S state has been initialized in the routing space using the methods provided in Ref. [14]. Green dots represent stabilizer measurements whose outcomes produce the result of the joint logical Pauli measurement.



(a) Initial arrangement. (b) First stage, d QEC rounds. (c) Second stage, $3d/2+4$ QEC rounds.

FIG. 14. Implementing a T gate on a logical patch. In (a), a patch in a T state has been provided in some additional space generated by a magic state factory using methods described in Sec. IV A. Green dots represent stabilizer measurements whose outcomes produce the result of the joint logical Pauli measurement. S correction is not shown, but occurs in (c) after the Pauli X measurement.

round, and the S correction requires $3d/2+3$ QEC rounds, leading to a total of $5d/2+4$ QEC rounds to implement a logical T gate.

Finally, it is worth noting that other sequences of gates can also be implemented using these techniques with no extra cost. In general, any sequence consisting of only $T/T^\dagger$, $S/S^\dagger$, and Z gates can be implemented using the protocol above at the cost of implementing a single T gate. Further details are provided in Appendix B.

C. Moving Clifford gates

In this section, we will consider another way of implementing the logical circuit from Sec. II on the surface code based on Ref. [12]. This technique offers the benefit of only needing to think about how to implement the non-Clifford gates, but at the cost of increasing the complexity of implementing such gates.

1. Pauli product rotations

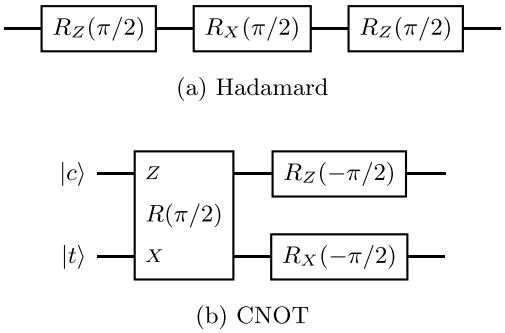
The key to this implementation method is that the logical gates we want to implement can be realized as rotations in a particular single- or multiqubit Pauli basis. More formally, an n-qubit quantum gate can be implemented as a sequence of rotations $R_{P_j}(\theta_j) = e^{-iP_j\theta_j/2}$ for suitably chosen $P_j \in \{I, X, Y, Z\}^{\otimes n}$ and θ_j.[1] The simplest example of this phenomenon is the Pauli gates themselves, which can be implemented as $P = R_P(\pi)$. Similarly, the T and S gates are both single-qubit rotations in the Z basis, and can thus be realized as $T = R_Z(\pi/4)$ and $S = R_Z(\pi/2)$, respectively. Single-qubit Pauli measurements, although not rotations around a Pauli basis, can also be seen as operations which project a state into a Pauli basis. In the case of QPE for example, measurements project a state into the Z basis.

The remaining gates to translate into this picture are the CNOT and Hadamard gates. Although not as easy to see as the gates listed above, both of these gates can be implemented as sequences of Pauli $\pi/2$ rotations given in Fig. 15 [12]. The Hadamard gate can be decomposed as $H = R_Z(\pi/2)\cdot R_X(\pi/2)\cdot R_Z(\pi/2)$, up to a global phase. The CNOT can be written as a joint $\pi/2$ $Z\otimes X$ rotation, followed by a $-\pi/2$ Z rotation on the control qubit, and a $-\pi/2$ X rotation on the target qubit. This is similar to the circuit used in Fig. 8, but with Pauli $\pi/2$ rotations rather than Pauli measurements.

2. Moving Pauli rotations

The benefit of describing operations as rotations in a Pauli basis is that it becomes easier to understand how to transform them without modifying the outcome of the circuit. For example, in Fig. 16, a $\pi/2$ rotation in the X basis is moved past a $\pi/4$ rotation in the Z basis. The result is that the Z rotation is transformed into a $\pi/4$ rotation in the $iXZ = i(-iY) = Y$ basis.

These transformations can be applied more generally as well, the rules for which we discuss in Appendix C. The benefit of these transformations to the circuit is that we can move all π and $\pi/2$ Pauli rotations, which correspond to Pauli and Clifford operations, past the final measurement operation of the circuit. Operations beyond this point do not affect the outcome of our circuit, and therefore do not need to be implemented. Thus we have reduced our circuit to only involving $\pi/4$ Pauli rotations, which correspond to a generalization of T gates, and joint Pauli measurements. We shall now look at how to implement these more general operations.



(a) Hadamard

(b) CNOT

FIG. 15. Implementing (a) Hadamard and (b) CNOT gates as Pauli $\pi/2$ rotations.

[1] An astute reader may notice that our definition of $R_{P_j}(\theta_j)$, and by extension our translation of Clifford and non-Clifford gates to Pauli rotations, differs from that in Ref. [12] by a factor of 2. This is to ensure correct periodicity, such that $R_P(\theta+2\pi) = R_P(\theta)\forall P, \theta$. This is also consistent with the definition of Pauli rotations in other texts such as, for example, Ref. [32].

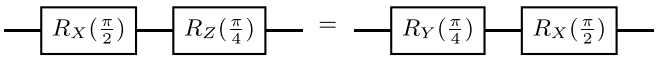


FIG. 16. Moving a $\pi/2$ X rotation past a $\pi/4$ Z rotation.

3. Implementing $\pi/4$ joint Pauli rotations

First we shall show how to reduce the $\pi/4$ joint Pauli rotations to joint Pauli measurements. These will then be implemented using a particular patch layout and lattice surgery operations in Sec. III C 4.

A circuit for implementing $\pi/4$ rotations is presented in Fig. 17. This can be seen as a generalization of the T gate circuit in Fig. 13, where now the single-qubit Z basis has been replaced with a general multiqubit basis P. The auxiliary qubit required for this operation is the same $|T\rangle = (|0\rangle + e^{i\pi/4}|1\rangle)/\sqrt{2}$ state from Sec. III B 4.

Because the rotation basis has generalized, so too have the corrective gates performed after the measurement. Now, instead of single-qubit S and Z gates we have more general $\pi/2$ and π rotations in an arbitrary Pauli basis P. The implementation of the π rotation is still a Pauli operation, and can be either tracked in software or implemented transversely as before. As for the $\pi/2$ rotation, one can account for this by employing the same techniques as described in Sec. III C 2 in an online fashion, moving the rotation past the final round of measurements to effectively remove it from the circuit and adjusting the subsequent operations accordingly [12].

4. Implementing joint Pauli measurements

Finally we discuss how to implement general Pauli measurements between patches on a surface code. The specific arrangement we use is given in Fig. 18(a). Note that this patch has more routing space than the one in 7, this is because the more general operations require access to both the horizontal and vertical observables of the patches. This results in six logical patches arranged on a grid of $(3d+4)\times(2d+2)$ data qubits, or $2(3d+4)\times(2d+2)$ physical qubits total.

The most challenging operations to implement are those which include the Y basis of a qubit. This is because the Y basis does not correspond to the horizontal or vertical observable on a surface code patch, but is instead a product of both the horizontal and vertical observables. One option is to decompose $\pi/4$ rotations which involve the Y basis of a qubit into a sequence of $\pi/4$ and $\pi/2$ rotations which only act on the X and Z bases [12]. However, doing so introduces $\pi/2$ rotations which cannot be moved past the $\pi/4$ rotation

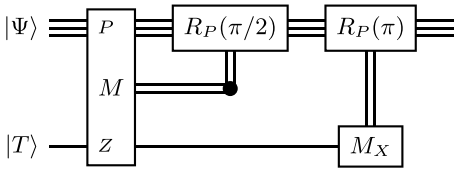


FIG. 17. Implementing a $\pi/4$ rotation on an n-qubit quantum state $|\Psi\rangle$ in a Pauli basis P using an auxiliary state $|T\rangle = (|0\rangle + e^{i\pi/4}|1\rangle)/\sqrt{2}$. This can be seen as a generalization of the T gate circuit in Fig. 13.

without reintroducing the Y basis, so such rotations would need to be implemented.

Instead, we utilize another technique from [13] to implement Y basis measurements directly via lattice surgery operations. Some example measurements for implementing Pauli $\pi/4$ rotations in the $Y\otimes X$ and $Z\otimes Y$ bases are given in Fig. 18. These joint measurements require d QEC rounds, followed by a single QEC round to measure the auxiliary patch in the X basis. These two sets of measurement results give us the corrections to move past future operations.

Here we utilize some lattice surgery techniques not used in Sec. III B. First, we add weight-five stabilizers, known as twist defects, which involve a Y Pauli term on one of the qubits. To ensure the surrounding stabilizers commute with the twist defects, we utilize two other lattice surgery techniques: first, we add domain walls, which are denoted by half-blue-half-grey squares and act as a combination of X and Z stabilizers; and second, we add elongated weight-four stabilizers, which are denoted by blue and grey rectangles. It is important to note that although these techniques allow for direct implementation of joint measurements involving the Y basis, there is an additional cost in that measuring these longer stabilizers requires additional connectivity compared to the layout used in Sec. III B. These extra connections between measurement qubits are not uniform, and shown by arrows in Fig. 18. In general, for distance d a total of $4d$ extra connections are required for implementing this algorithm, which connect four columns of adjacent measurement qubits.

IV. ERROR CORRECTION OVERHEADS

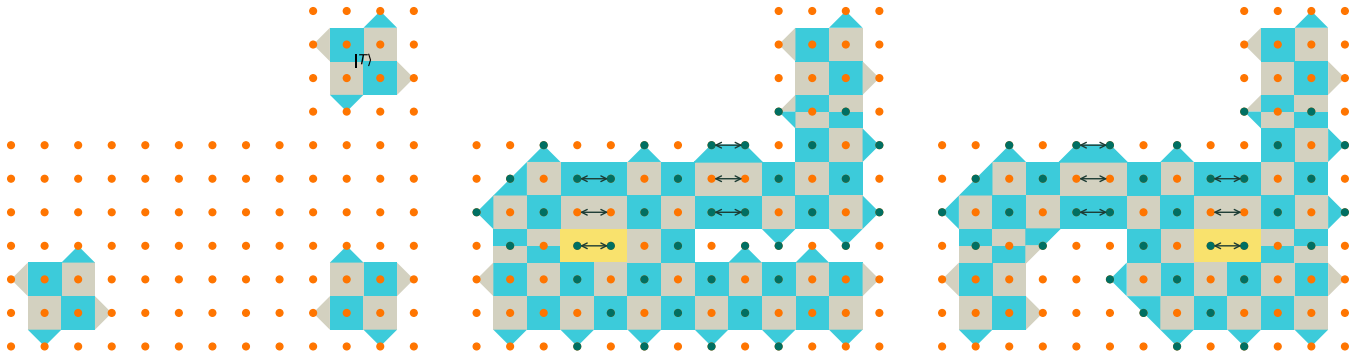
We are now ready to discuss the cost of implementing these logical gates on the surface code. There are two primary sources of error which contribute to the probability of a failure at the error-correction level: first, errors from generating $|T\rangle$ states, which we shall explore in Sec. IV A; and second, errors from a logical failure on a qubit, which we shall explore in Sec. IV B.

A. Generating $|T\rangle$ states

Both of the methods used in Sec. III require additional qubits initialized in the $|T\rangle$ state. It is possible to initialize a surface code patch into an arbitrary qubit state $|\psi\rangle$, by initialising one data qubit of the patch in the $|\psi\rangle$ state, followed by d rounds of measurements [9]. However, initialising a data qubit into an arbitrary state means that this qubit is initially unprotected from errors, so this method cannot be implemented in a way that reduces the logical error probability below the physical error probability. In fact, it can be shown that there is no fault-tolerant way of initialising nonstabilizer states such as the $|T\rangle$ state on the surface code.[2]

Even though patches cannot be initialized in the $|T\rangle$ state in a way that suppresses errors, it is possible to use distillation protocols to reduce the error probability of $|T\rangle$ states.

[2]Note that there are other ways of initialising a $|T\rangle$ state on the surface code which are more immune to errors, however these rely on post-selection and therefore might introduce additional overheads. For simplicity we shall not focus on this method.



(a) Initial layout. (b) Pauli $Y \otimes X \otimes Z$ measurement. (c) Pauli $Z \otimes Y \otimes Z$ measurement.

FIG. 18. Layout of logical patches for implementing joint Pauli measurements. In (a), the two qubits used in the logical circuit are at the bottom, and an auxiliary qubit is initialized in the $|T\rangle$ state at the top. Example joint measurements required for implementing $\pi/4$ Pauli $Y \otimes X$ and $Z \otimes Y$ operations in $d+1$ QEC rounds are presented in (b) and (c), respectively. The auxiliary $|T\rangle$ state is always measured in the Z basis as part of the joint measurement. Green dots represent stabilizer measurements whose outcomes produce the result of the joint logical Pauli measurement. Twist defects are presented in yellow. Arrows between neighboring measurement qubits show extended connectivity than what is required for the methods presented in Sec. III B.

These protocols take multiple noisy $|T\rangle$ states and output a smaller number of $|T\rangle$ states with a reduced error probability [12,17–19,50–53]. For example, if it is possible to generate 15 $|T\rangle$ states each with error probability p, it is possible to distill these into a single $|T\rangle$ state with error probability $35p^3$ [17]. It is also possible to concatenate these factories to reduce the error probability even further. For example, if the 15-to-1 protocol is used to generate 15 $|T\rangle$ states each with error probability $35p^3$, these can then be used in another 15-to-1 protocol to generate a single $|T\rangle$ state with error probability $35(35p^3)^3 = 1,500,625p^9$ [12]. The cost with these protocols is that reducing the error probability requires additional resources in terms of both time and number of qubits. A summary of several protocols and their associated costs is provided in Ref. [19]. We also provide some example resource estimates for 15-to-1 factories in Table I, generated using code from Ref. [19].

When choosing a suitable protocol, there are multiple factors that we need to consider. First, we need to consider the overall logical failure probability from faulty $|T\rangle$ state generation. This means that if our logical circuit uses m T gates—and therefore requires m $|T\rangle$ states—we need to choose a probability of distilled state failure p_{dist} such that $m \times p_{\text{dist}}$ is within our error bounds.

The second aspect we need to consider is the time required to generate each $|T\rangle$ state. In order to avoid logical qubits remaining idle as we wait for $|T\rangle$ states to be generated, we need to ensure that $|T\rangle$ states are generated fast enough that they are available as and when they are needed. This depends on both the number of QEC rounds required to generate the $|T\rangle$ states, but also the number of QEC rounds required to implement these logical operations. If we implement Clifford and T gates directly as described in Sec. III B, the circuit primarily consists of alternating sequences of Hadamard gates, which take $3d+4$ QEC rounds, and T-like gates, which take between $d+1$ and $5d/2+4$ QEC rounds, depending on whether or not an S gate correction is required. This means that when implementing Clifford and T gates directly, a $|T\rangle$ state needs to be produced at least once every $4d+5$ QEC rounds. In comparison, when Clifford operations have been moved through the circuit as described in Sec. III C, the only operations required are a single joint Pauli measurement and a single X basis measurement, meaning that a $|T\rangle$ state must be produced every $d+1$ QEC rounds. If a single distillation protocol cannot generate states fast enough, multiple instances of the protocol can be run in parallel to generate states more frequently, at the cost of increasing the number of physical qubits [12]. As we show in Appendix D, up to four factories can be placed around the two corners at the top of the routing space. It is possible to add even more factories beyond these four, but doing so could require additional space for routing and storage of $|T\rangle$ states. On the other hand, if a logical $|T\rangle$ state can be generated faster than required, additional storage space is required to protect the state from errors while it waits to be consumed, which can be included as part of the routing space estimates.

TABLE I. Resource estimates for some example 15-to-1 $|T\rangle$ state factories at physical error rates 10^{-3} and 10^{-4}.

Physical error probability	10^{-3}	10^{-4}
X error distance	11	5
Z error distance	5	3
Measurement error distance	5	3
Distillation error	8.66×10^{-6}	4.68×10^{-6}
Number of qubits	2066	522
Expected number of QEC rounds	31.30	18.05

B. Estimating code distance

To reduce the probability of a logical error occurring on one of our logical qubits, we can tweak the code distance d. A higher distance will reduce the probability of getting a sequence of physical errors which lead to a logical error, but comes at the cost of increasing both the number of physical qubits per logical qubit, and the number of QEC rounds per

TABLE II. Detailed resource estimates required to perform the iterative QPE circuit described in the main text for the hydrogen molecule, considering physical error rates of 10^{-3} and 10^{-4}. Resource estimates for the $|T\rangle$ state factories are in Table I.

Implementation method	Direct implementation		Move Cliffords	
Physical error rate	10^{-3}	10^{-4}	10^{-3}	10^{-4}
Code distance	12	6	11	5
Number of qubits for logical circuit	1,352	392	1,776	456
Probability of logical error on any logical qubit	4.8×10^{-3}	8.6×10^{-4}	4.2×10^{-3}	2.3×10^{-3}
Number of QEC rounds between non-Clifford gates	53	29	12	6
Number of $\|T\rangle$ state factories	1	1	3	4
Number of qubits for generating $\|T\rangle$ states	2,066	522	6,198	2,088
Number of qubits for storing $\|T\rangle$ states	288	72	726	200
Probability of $\|T\rangle$ state error	3.1×10^{-3}	1.8×10^{-3}	3.1×10^{-3}	1.8×10^{-3}
Total number of physical qubits	3,706	986	8,700	2,744
Number of QEC rounds	31,179	17,271	4,665	2,331
Total error probability	8.1×10^{-3}	2.6×10^{-3}	7.3×10^{-3}	4.1×10^{-3}

logical operation. In the case of the surface code, the probability of a logical error on a single logical qubit per code cycle assuming a depolarising noise model can be estimated as

$$p_L(p, d) = 0.1(100p)^{(d+1)/2}, \tag{25}$$

where p is the physical error probability [10,12,52]. For the purpose of this application, we want to choose a sufficiently high d that the probability of a logical error occurring on any qubit during any QEC round is within our error bound. We use Eq. (25) to approximate our probability of a logical error at any point in the computation as

$$(n_{\text{data}} + n_{\text{route}}) \times n_{\text{meas}} \times p_L(p, d), \tag{26}$$

where n_{data} is the number of surface code patches for our data qubits, n_{route} is the number of additional patches used for routing,[3] and n_{meas} is the number of QEC rounds. Given these parameters and physical error probability p, we can pick a distance by choosing an appropriate d such that Eq. (26) is within our target failure probability.

C. Results

We are now ready to estimate error correction overheads for our iterative quantum phase estimation circuit. As a recap, our circuit consists of 13 X gates, 169 Z gates, 34 CNOT gates, 411 Hadamard gates, 13 $S/S^\dagger$ gates, 386 $T/T^\dagger$ gates, and three Z basis measurements. As previously described, X and Z gates are free as they can be implemented transversely at the start of a QEC round. Of the S and $S^\dagger$ gates, one is used in a sequence of T gates, and can therefore be implemented as a T-like gate. This leaves our costing as 411 Hadamard gates, 34 CNOT gates, 386 T-like gates, 12 $S/S^\dagger$ gates, and three measurements.

[3]Note that at various points of the computation these routing patches are unused. This means that errors occurring on them will not lead to an overall failure. However, in the worst case a logical error occurs on one of these patches while it is in use, which can in turn lead to a failure of the overall computation, hence why we include both data patches and routing patches in this calculation.

We also need to make assumptions on the error correction requirements of our algorithm. We assume physical errors correspond to depolarising noise with a physical error probability ranging between 10^{-4} and 2×10^{-3}. We also assume a target failure probability of 1%, though a higher target probability can be used to reduce overheads [7,18]. This target failure probability is split evenly, so the probability of errors occurring from faulty $|T\rangle$ state preparation is at most 0.5%, and the probability of logical errors happening on the qubits used in the logical circuit is also at most 0.5%. For 386 T-like gates, the required error rate per T gate in order to meet this error budget is 1.3×10^{-5}. This is a higher error probability than what is seen from many distillation techniques [18,19], so instead we use code from [19] to look for smaller factories which still fit within our target failure probability. Note that both factories presented in Table I suffice at error rates 10^{-3} and 10^{-4}.

Our results are presented in Fig. 1. To help explain these resource estimates, the rest of this section will provide detailed costings for physical error rates of 10^{-3} and 10^{-4}. These physical error rates are commonly used when estimating the resource requirements of fault-tolerant quantum algorithms [4,12,19]. For ease of reading, a summary of these results is presented in Table II.

1. Cost of directly implementing Clifford and T gates

Using the estimates described in Sec. III B, we note that there are four logical patches to consider when estimating code distance. In terms of time requirements, CNOT and Hadamard gates require $3d + 4$ rounds, $S/S^\dagger$ gates require $3d/2 + 3$ rounds, T-like gates require up to $5d/2 + 4$ rounds, and Z basis measurements require a single round. This brings our total number of rounds to $2318d + 3363$.

Using Eq. (26), we find that for a physical error probability of 10^{-3}, distance $d = 12$ achieves a logical error probability of 3.9×10^{-3}, requiring 1,352 physical qubits for the patches and 31,179 QEC rounds. The factory in Table I produces a $|T\rangle$ state with error probability 8.1×10^{-6} on average once every 31.3 QEC rounds, meaning a single factory is sufficient. This factory uses 2066 physical qubits, along with 288 physical qubits for storing $|T\rangle$ states. Combined with our 1352 qubits

for the logical circuit and routing, this leads to a total of 3706 physical qubits. The additional logical qubit for storing $|T\rangle$ states increases the probability of a logical error to 4.9×10^{-3}, leading to a total error probability of 8.1×10^{-3}.

For a physical error probability of 10^{-4}, an error probability of 6.9×10^{-4} can be achieved with distance $d = 6$, which requires 14 953 QEC rounds and 288 physical qubits. A $|T\rangle$ state needs to be generated every 29 rounds with an error probability of 1.3×10^{-5}. For this physical error probability, the factory in Table I produces a $|T\rangle$ state on average every 18.05 rounds with error probability 4.7×10^{-6}. We use a single factory, which requires 522 physical qubits, along with 72 physical qubits for storing $|T\rangle$ states. Adding in our 392 qubits for the logical circuit and routing, this gives us a total of 986 physical qubits. The additional storage space for data qubits increases the probability of a logical error on qubits used in the quantum circuit to 8.6355×10^{-4}, leading to a total error probability of 2.6×10^{-3}.

2. *Cost of moving Clifford gates*

If we choose to move Clifford gates through the circuit, we are left with a total of 386 Pauli $\pi/4$ rotations, each of which requires $d + 1$ QEC rounds, and three joint Pauli measurements, which require d QEC rounds each. Therefore our total number of QEC rounds is $389d + 386$. We also have six logical patches allocated for both the logical circuit and routing.

For a physical error probability of 10^{-3}, distance $d = 11$ achieves a logical error probability of 2.8×10^{-3}, requiring 1,776 physical qubits and 4,665 QEC rounds. A $|T\rangle$ state needs to be produced once every 12 QEC rounds. We use the same 15-to-1 factory as in Table I, however a single factory is not sufficient for producing one $|T\rangle$ state every 12 rounds. Instead, we use three factories, which produce a single $|T\rangle$ state on average once every 10.4 QEC rounds and require 6,198 physical qubits for implementing the factories. The additional three logical qubits for storing $|T\rangle$ states increase the probability of a logical error on a qubit used in the quantum circuit to 4.2×10^{-3}, leading to a total error probability of 7.3×10^{-3}. The total number of physical qubits is 8700.

For a physical error probability of 10^{-4}, distance $d = 5$ achieves a logical error probability of 1.4×10^{-3} at a cost of 2331 QEC rounds and 456 physical qubits. Using the same factory as in Table I produces a $|T\rangle$ state with sufficiently low failure probability every 18.05 rounds, but a $|T\rangle$ is required every 6 rounds. Arranging four factories around the data qubits is sufficient to remove this bottleneck. These four factories require 2088 physical qubits, and 200 physical qubits for storing $|T\rangle$ states. Adding this to our 456 physical qubits for the logical patches and routing leads to a total of 2744 physical qubits. The extra four logical qubits for storing $|T\rangle$ states increase the probability of a logical error on a qubit used in the quantum circuit to 2.3×10^{-3}, which means the total error probability is 4.1×10^{-3}.

D. Analysis

As we can see from Fig. 1, there are still some significant overheads introduced from quantum error correction. The most optimistic error rates still require hundreds of physical qubits and thousands of QEC rounds, while at an error rate of 0.2% this circuit requires tens of thousands of qubits and QEC rounds.

From the detailed costings of Sec. IV C, we can identify several bottlenecks with these approaches. For physical qubits, the overhead mostly comes from $|T\rangle$ state factories: at a physical error rate $p = 10^{-4}$ a single factory requires 522 physical qubits, nearly twice as many as required by the data qubits when implementing Clifford and T gates directly. This is even more prominent when moving Clifford gates through the circuit, where of the 2,744 physical qubits required at error rate 10^{-4}, 2,288 are for preparing and storing $|T\rangle$ states. Although using fewer factories can reduce the number of physical qubits, this creates time bottleneck as the data qubits need to remain idle while $|T\rangle$ states are prepared.

Although it is expected that the overhead from such factories will become a less significant factor as we move towards larger quantum computations [19], for early fault-tolerant quantum circuits these overheads are likely to be more costly. This could be improved via more efficient small footprint factories like the ones presented in Ref. [19], as well as the use of error-mitigated T gates [16].

When moving Clifford operations through the circuit, another space overhead comes from joint Pauli operations in the Y basis. These require additional routing space and extra connectivity. Optimising the circuit to remove such measurements would also therefore reduce the routing overhead.

In time complexity, a significant bottleneck is the long sequences of Hadamard and T gates which come from GRIDSYNTH decompositions. Of the 846 logical operations implemented in this circuit, 797 are either Hadamard or T gates. The Hadamard gate is especially expensive, requiring $3d + 4$ QEC rounds. In practice, this means that more than half the QEC rounds are spent implementing Hadamard gates: at a physical error rate of 10^{-4}, 9,042 of the 17 271 QEC rounds are spent implementing logical Hadamard gates. Time requirements can also be further reduced in general by using gate-based teleportation to execute gates in parallel, though this comes at a cost of more physical qubits [12].

Finally it is worth emphasising that there are other ways in which these resource estimates can be improved above the quantum error correction layer, such as the use of different quantum algorithms [54] and decomposition techniques [47].

V. CONCLUSION

As we enter the era of early fault-tolerant quantum computers, where quantum error correction is able to suppress errors on a logical qubit and basic logical gates are demonstrable, it is essential for us to understand the progress required for large-scale fault-tolerant quantum algorithms. Understanding the requirements of small applications is an important step in the process. In this work, we have analysed a minimal application: estimating the ground-state energy of the hydrogen molecule. We have used several techniques to reduce the estimated resources to approximately 900 physical qubits and 15 000 QEC rounds through implementing Clifford and T operations directly, and approximately 2700 physical qubits and 2331 QEC rounds when implementing general Pauli $\pi/4$ rotations.

It is worth emphasising that even for this small application, the numbers of physical qubits and gates required is several orders of magnitude larger than what has been performed experimentally so far. There are a number of further optimizations which can be made across the quantum computing stack in the hope of reducing these estimates. At the algorithmic level, techniques such as qubitization have been shown to produce asymptotically shorter quantum circuits [55–58], and could potentially offer improvements even for this minimal example [54]. Statistical phase estimation methods can allow reduced circuit depth in exchange for performing more samples [22,23,40], and are often stated as being particularly appropriate for the early fault-tolerant era for this reason. At the gate synthesis level, alternative techniques have produced circuits with a smaller T count, at the cost of additional logical qubits [47]. When implementing $\pi/4$ joint Pauli rotations, the number of QEC rounds can be further reduced by implementing noncommuting rotations in parallel on separate patches before using teleportation to combine them, though this comes at a cost of more physical qubits [12]. Finally, improvements can be made to the implementation of non-Clifford gates which are more targeted towards early fault-tolerant quantum devices, such as the use of error mitigation when implementing faulty T gates [16], avoiding the need for magic state distillation factories. Algorithms such as statistical phase estimation may remain well suited even in the presence of error mitigation [26].

A final note is that these estimates assume that quantum computers are affected specifically by depolarising noise [7,10,12]. While depolarising noise is easy to mathematically model, the physical noise that affects real-world devices is more complex and cannot necessarily be captured by such a model. An important direction of future work is investigating other more realistic noise models such as leakage and deriving similar scaling formulas to that presented in Eq. (25).

The source code for generating and running the logical circuits, and estimating resources, is available on GitHub [48].

ACKNOWLEDGMENTS

We thank Ophelia Crawford, Earl T. Campbell, Nicole Holzmann, Jacob M. Taylor, and other Riverlane colleagues for insightful discussions, and Daniel Litinski for making his code for estimating the resource requirements of $|T\rangle$ factories publicly available.

APPENDIX A: LOGICAL CIRCUIT FIGURES

The complete logical circuits for estimating the ground-state energy of the hydrogen molecule using textbook QPE

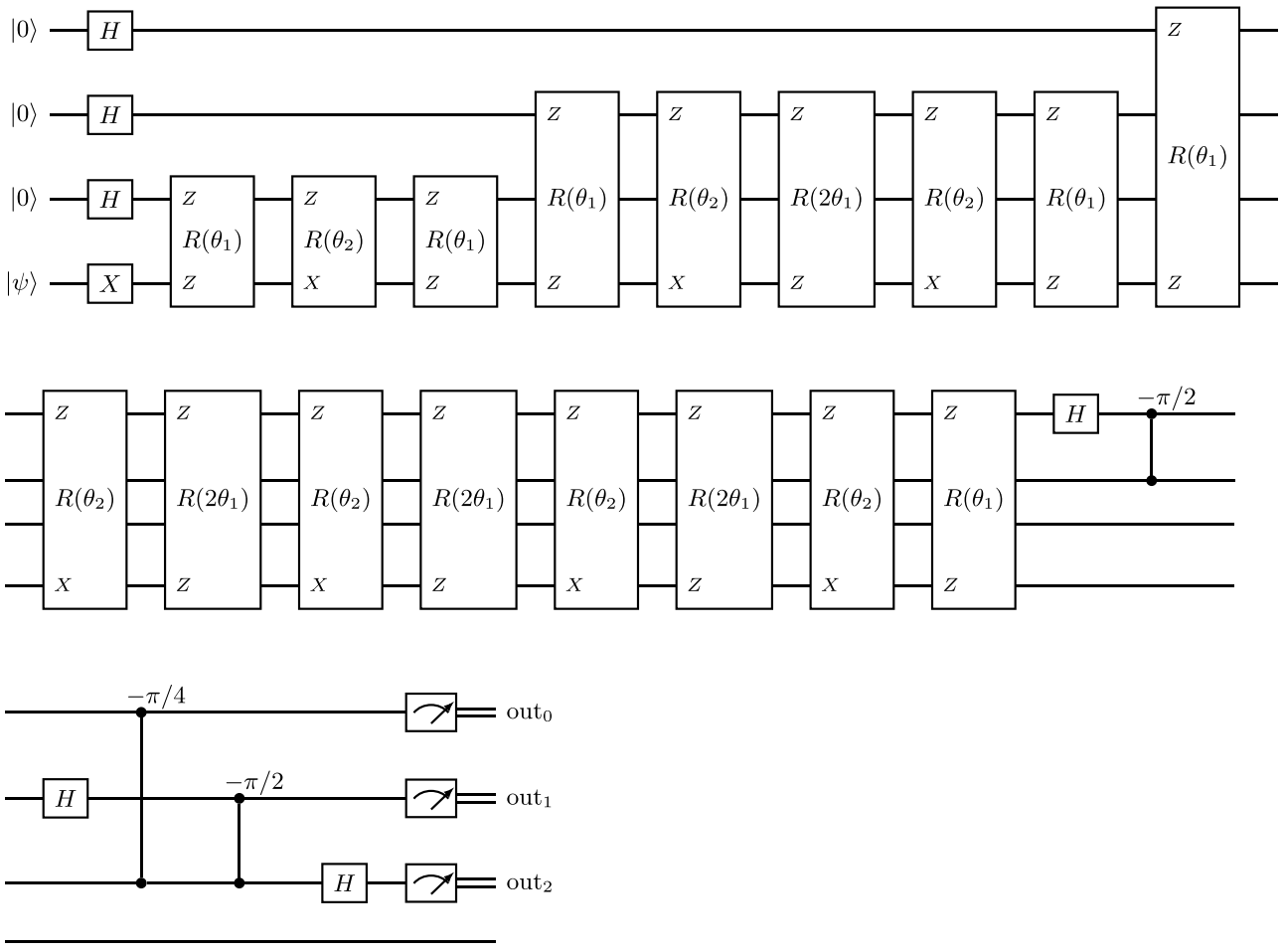


FIG. 19. Logical quantum circuit for textbook QPE to three bits of precision.

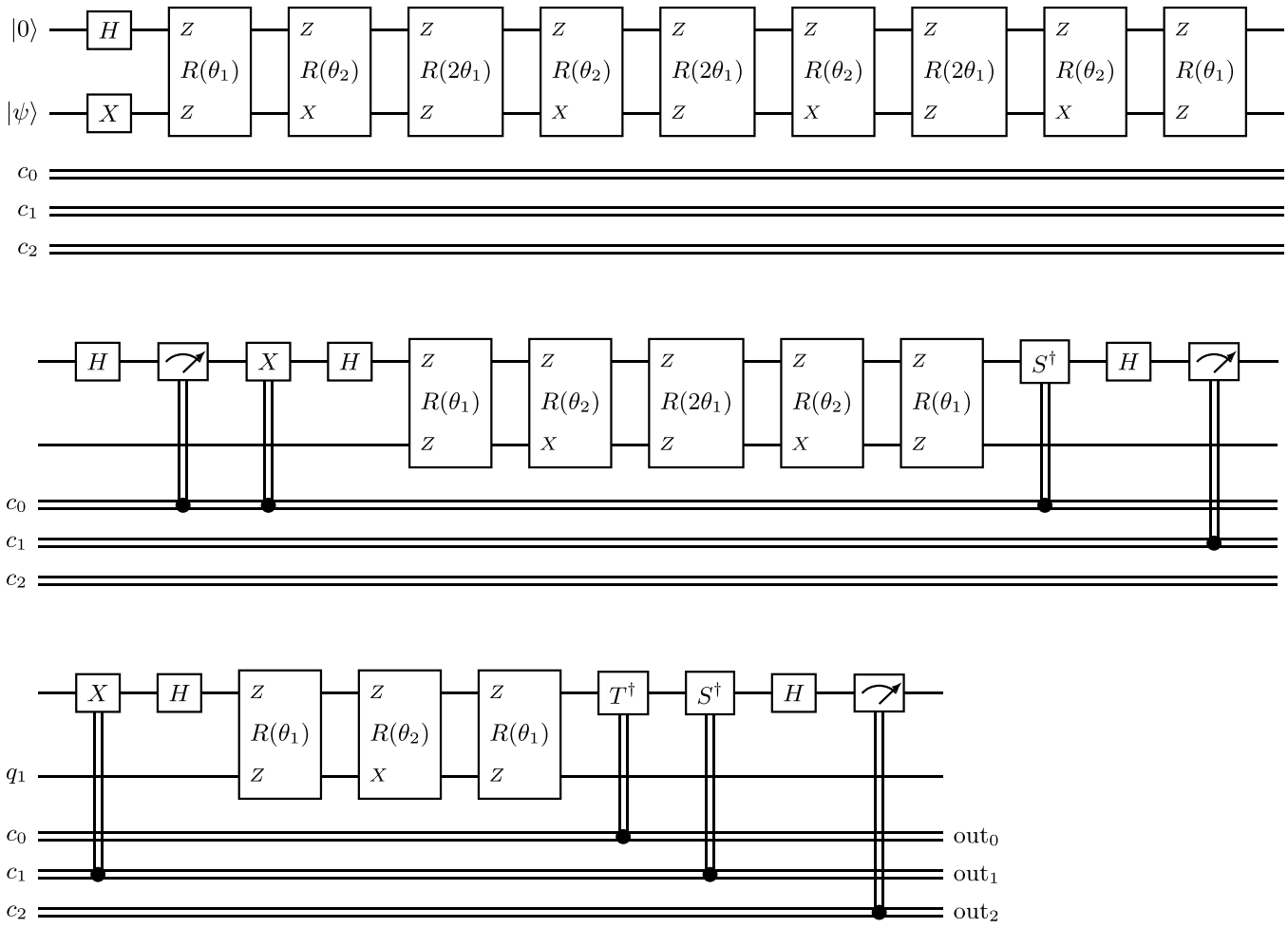
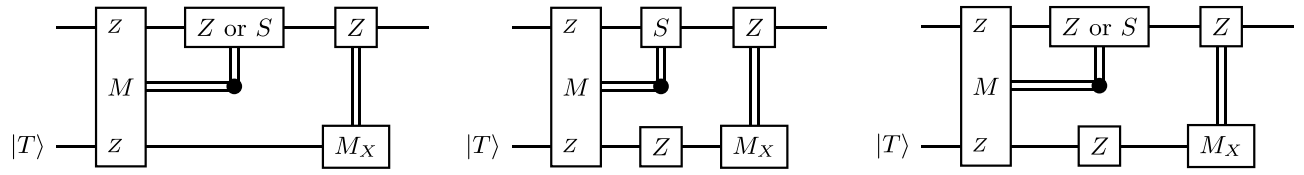


FIG. 20. Logical quantum circuit for iterative QPE to three bits of precision.

and iterative QPE are given in Figs. 19 and 20, respectively. In the iterative QPE circuit, the classically controlled X gates are used to reset the measured qubits to the $|0\rangle$ state.

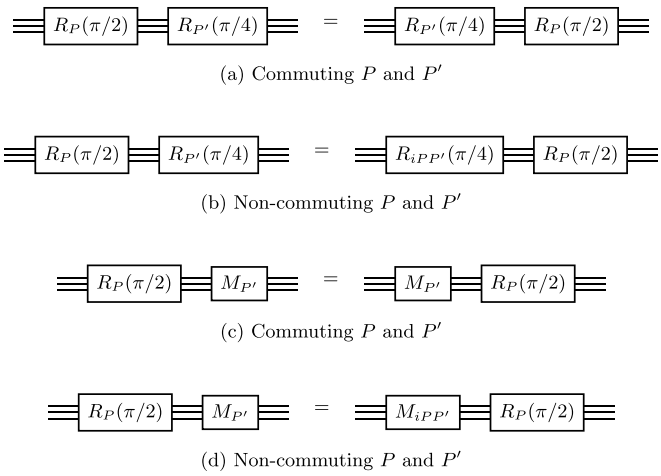
APPENDIX B: IMPLEMENTATION OF T-LIKE GATES

Here we explain how to implement any sequence of $T/T^\dagger$, $S/S^\dagger$ and Z gates. This is done in three steps: first, by removing inverse gates by noting that $S^\dagger = ZS$ and $T^\dagger = ZST$; second, by combining the different phase gates into a sequence consisting of at most one Z gate, one S gate, and one T gate; and third, by absorbing the non-T gates into the conditional operations in Fig. 13. In Figure 21(a), we give an example of implementing a T gate followed by an S gate using this procedure: we combine the S gate following the T gate with the conditional S gate, to create a gate where either a Z operation is applied at no extra cost, or an S gate is applied, depending on the joint $Z \otimes Z$ measurement outcome. Similar circuits can also be generated for TZ and TSZ gate sequences, as shown in Fig. 21(b) and 21(c), respectively. Note that the



(a) T gate followed by an S gate (b) T gate followed by a Z gate (c) T gate followed by an S gate and a Z gate

FIG. 21. Example circuits for implementing T-like gates through joint $Z \otimes Z$ measurements. In (a), a T gate followed by an S gate is implemented with the same cost as computing a single T gate as shown in Fig. 13, by adjusting the Clifford operations performed after the measurement. Note that in (a), either a Z gate is implemented or an S gate is implemented, depending on the outcome of the joint $Z \otimes Z$ measurement. In (b) and (c), we use the same techniques for implementing a T gate followed by a Z gate, and a T gate followed by an S gate and a Z gate, respectively. All circuits require an additional qubit initialized in the $|T\rangle = (|0\rangle + e^{i\pi/2}|1\rangle)/\sqrt{2}$ state.



(a) Commuting P and P'

(b) Non-commuting P and P'

(c) Commuting P and P'

(d) Non-commuting P and P'

FIG. 22. Moving n-qubit $\pi/2$ Pauli rotations past other Pauli operations. (a) A $\pi/2$ rotation in Pauli basis P can be swapped with a $\pi/4$ rotation in a commuting Pauli basis P'. (b) A $\pi/2$ rotation in a Pauli basis P can also be swapped with a $\pi/4$ rotation in a noncommuting basis P', by modifying the basis of the $\pi/4$ rotation to iPP'. These rules also apply to moving $\pi/2$ rotations past Pauli measurements, as shown in (c) and (d).

Z operation on the auxiliary qubit in Figures 21(b) and 21(c) can be implemented at no extra cost by simply inverting the result of the X basis measurement.

APPENDIX C: TRANSLATION RULES FOR PAULI ROTATIONS

Here we explain the rules for manipulating Pauli rotations used in Sec. III C. In general, a $\pi/2$ rotation in some Pauli basis P can be moved past a $\pi/4$ operation in some Pauli basis P' without modification to either basis if P and P' commute, or by modifying P' to iPP' if they do not commute. These rules are presented graphically in Fig. 22.[4] These rules can also be used to move π rotations as well, by noting that $R_P(\pi) = R_P(\pi/2) \cdot R_P(\pi/2)$.

It is important however to note that these remaining non-Clifford operations can be more complicated than the single-qubit non-Clifford operations in the original logical circuit. An example of this is shown in Fig. 23, where a $\pi/2$ $Z \otimes X$ rotation, such as the one that shows up in the CNOT circuit of Fig. 15(b), is moved past a $\pi/4$ Z rotation on the second qubit. Noting that $Z \otimes X$ does not commute with $I \otimes Z$, the basis for the $\pi/4$ rotation becomes

$$i(Z \cdot I) \otimes (X \cdot Z) = iZ \otimes (-iY) = Z \otimes Y,$$

thus what was originally a $\pi/4$ rotation across a single qubit has now become a $\pi/4$ rotation across multiple qubits.

[4]Note that our rules for moving Pauli rotations past measurements are slightly different from those presented in Ref. [12]. This is because unlike the circuits in Ref. [12] where there is a single layer of measurement gates at the end of the circuit, the iterative QPE circuit in Fig. 2(b) features mid-circuit measurements and therefore other computations happen after a measurement gate.

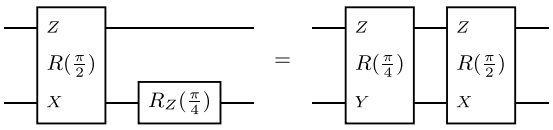
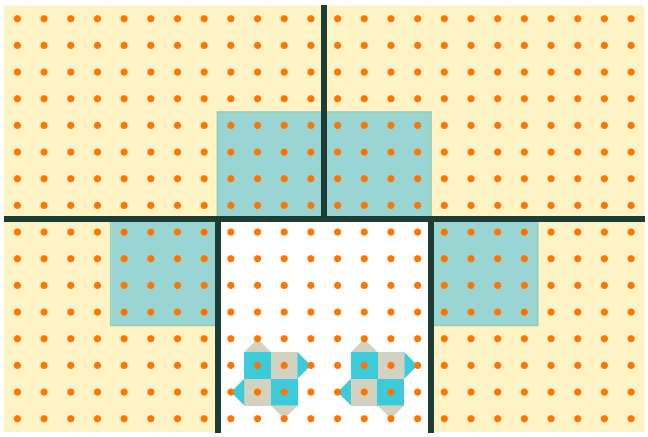


FIG. 23. Moving a two-qubit $\pi/2$ $Z \otimes X$ rotation past a single-qubit $\pi/4$ Z rotation leads to a two-qubit $\pi/4$ $Z \otimes Y$ rotation.

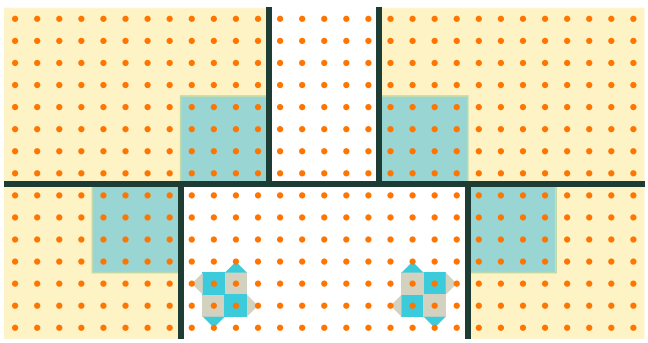
APPENDIX D: ARRANGEMENT OF $|T\rangle$ STATE FACTORIES

In Fig. 24, we show how to arrange magic state factories around the logical patches and routing space for both implementing Clifford and T gates directly and moving Clifford gates through the circuit. In both cases, it is easily possible to arrange up to four factories around the logical patches. It is also possible to arrange even more factories, however this might come at the cost of additional routing space. For simplicity we stick with up to four factories.

Note that in Fig. 24(b) there is some unused space at the top of the arrangement. This space has been left empty for



(a) Directly implementing Clifford and T gates



(b) Moving Clifford gates

FIG. 24. Arrangements of $|T\rangle$ state factories around the routing spaces for (a) implementing Clifford and T operations directly and (b) commuting Clifford operations. Green space denotes storage space for $|T\rangle$ states produced by the factories, which are denoted in yellow. Note that the full factories are not shown due to size.

ease of symmetry with the arrangement, but could be used as additional factory space. Such unused qubits are not included as part of the resource estimates presented in Sec. IV.

We also stress that the space in yellow is not the full space required for the factories, but simply an indicator of what space the factories can be placed in. The green lines denoting the boundaries between the factories and routing space should be seen as extending beyond the limits of Fig. 24, to however much space is required for individual factories.

[1] R. Acharya, I. Aleiner, R. Allen, T. I. Andersen, M. Ansmann, F. Arute, K. Arya, A. Asfaw, J. Atalaya, R. Babbush, D. Bacon, J. C. Bardin, J. Basso, A. Bengtsson, S. Boixo, G. Bortoli, A. Bourassa, J. Bovaird, L. Brill, M. Broughton *et al.*, Suppressing quantum errors by scaling a surface code logical qubit, Nature (London) **614**, 676 (2023).

[2] M. Reiher, N. Wiebe, K. M. Svore, D. Wecker, and M. Troyer, Elucidating reaction mechanisms on quantum computers, Proc. Natl. Acad. Sci. **114**, 7555 (2017).

[3] J. Lee, D. W. Berry, C. Gidney, W. J. Huggins, J. R. McClean, N. Wiebe, and R. Babbush, Even more efficient quantum computations of chemistry through tensor hypercontraction, PRX Quantum **2**, 030305 (2021).

[4] N. S. Blunt, J. Camps, O. Crawford, R. Izsák, S. Leontica, A. Mirani, A. E. Moylett, S. A. Scivier, C. Sünderhauf, P. Schopf, J. M. Taylor, and N. Holzmann, Perspective on the current state-of-the-art of quantum computing for drug discovery applications, J. Chem. Theory Comput. **18**, 7001 (2022).

[5] I. H. Kim, Y.-H. Liu, S. Pallister, W. Pol, S. Roberts, and E. Lee, Fault-tolerant resource estimate for quantum chemical simulations: Case study on Li-ion battery electrolyte molecules, Phys. Rev. Res. **4**, 023019 (2022).

[6] A. V. Ivanov, C. Sünderhauf, N. Holzmann, T. Ellaby, R. N. Kerber, G. Jones, and J. Camps, Quantum computation for periodic solids in second quantization, Phys. Rev. Res. **5**, 013200 (2023).

[7] C. Gidney and M. Ekerå, How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits, Quantum **5**, 433 (2021).

[8] D. Litinski, How to compute a 256-bit elliptic curve private key with only 50 million Toffoli gates, arXiv:2306.08585 [quant-ph].

[9] D. Horsman, A. G. Fowler, S. Devitt, and R. V. Meter, Surface code quantum computing by lattice surgery, New J. Phys. **14**, 123011 (2012).

[10] A. G. Fowler and C. Gidney, Low overhead quantum computation using lattice surgery, arXiv:1808.06709 [quant-ph].

[11] D. Litinski and F. v. Oppen, Lattice surgery with a twist: Simplifying Clifford gates of surface codes, Quantum **2**, 62 (2018).

[12] D. Litinski, A game of surface codes: Large-scale quantum computing with lattice surgery, Quantum **3**, 128 (2019).

[13] C. Chamberland and E. T. Campbell, Circuit-level protocol and analysis for twist-based lattice surgery, Phys. Rev. Res. **4**, 023090 (2022).

[14] C. Gidney, Inplace access to the surface code y basis, arXiv:2302.07395 [quant-ph].

[15] G. Watkins, H. M. Nguyen, V. Seshadri, K. Watkins, S. Pearce, H.-K. Lau, and A. Paler, A high performance compiler for very large scale surface code computations, arXiv:2302.02459 [quant-ph].

[16] C. Piveteau, D. Sutter, S. Bravyi, J. M. Gambetta, and K. Temme, Error mitigation for universal gates on encoded qubits, Phys. Rev. Lett. **127**, 200505 (2021).

[17] S. Bravyi and A. Kitaev, Universal quantum computation with ideal Clifford gates and noisy ancillas, Phys. Rev. A **71**, 022316 (2005).

[18] C. Gidney and A. G. Fowler, Efficient magic state factories with a catalyzed $|CCZ\rangle$ to $2|T\rangle$ transformation, Quantum **3**, 135 (2019).

[19] D. Litinski, Magic state distillation: Not as costly as you think, Quantum **3**, 205 (2019).

[20] E. T. Campbell, Early fault-tolerant simulations of the Hubbard model, Quantum Sci. Technol. **7**, 015007 (2022).

[21] L. Lin and Y. Tong, Heisenberg-limited ground-state energy estimation for early fault-tolerant quantum computers, PRX Quantum **3**, 010318 (2022).

[22] G. Wang, D. Stilck-França, R. Zhang, S. Zhu, and P. D. Johnson, Quantum algorithm for ground state energy estimation using circuit depth with exponentially improved dependence on precision, Quantum **7**, 1167 (2023).

[23] Z. Ding and L. Lin, Even shorter quantum circuit for phase estimation on early fault-tolerant quantum computers with applications to ground-state energy estimation, PRX Quantum **4**, 020331 (2023).

[24] S. Wang, S. McArdle, and M. Berta, Qubit-efficient randomized quantum algorithms for linear algebra, arXiv:2302.01873 [quant-ph] [PRX Quantum (to be published)].

[25] T. M. Graham, Y. Song, J. Scott, C. Poole, L. Phuttitarn, K. Jooya, P. Eichler, X. Jiang, A. Marra, B. Grinkemeyer, M. Kwon, M. Ebert, J. Cherek, M. T. Lichtman, M. Gillette, J. Gilbert, D. Bowman, T. Ballance, C. Campbell, E. D. Dahl *et al.*, Multi-qubit entanglement and algorithms on a neutral-atom quantum computer, Nature (London) **604**, 457 (2022).

[26] N. S. Blunt, L. Caune, R. Izsák, E. T. Campbell, and N. Holzmann, Statistical phase estimation and error mitigation on a superconducting quantum processor, PRX Quantum **4**, 040341 (2023).

[27] D. Wecker, M. B. Hastings, N. Wiebe, B. K. Clark, C. Nayak, and M. Troyer, Solving strongly correlated electron models on a quantum computer, Phys. Rev. A **92**, 062318 (2015).

[28] I. D. Kivlichan, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, W. Sun, Z. Jiang, N. Rubin, A. Fowler, A. Aspuru-Guzik, H. Neven, and R. Babbush, Improved fault-tolerant quantum simulation of condensed-phase correlated electrons via Trotterization, Quantum **4**, 296 (2020).

[29] S. McArdle, E. Campbell, and Y. Su, Exploiting fermion number in factorized decompositions of the electronic structure hamiltonian, Phys. Rev. A **105**, 012403 (2022).

[30] N. J. Ross and P. Selinger, Optimal ancilla-free Clifford+t approximation of z-rotations, Quantum Inf. Comput. **16**, 901 (2016).

[31] A. Y. Kitaev, Quantum measurements and the abelian stabilizer problem, arXiv:quant-ph/9511026.

[32] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, Cambridge, UK, 2010).

[33] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, Quantum algorithms revisited, Proc. R. Soc. Lond. A. **454**, 339 (1998).

[34] M. Mosca and A. Ekert, The hidden subgroup problem and eigenvalue estimation on a quantum computer, in *Quantum Computing and Quantum Communications*, edited by C. P. Williams (Springer, Berlin, Heidelberg, 1999), pp. 174–188.

[35] A. M. Childs, J. Preskill, and J. Renes, Quantum information and precision measurement, J. Mod. Opt. **47**, 155 (2000).

[36] M. Dobšíček, G. Johansson, V. Shumeiko, and G. Wendin, Arbitrary accuracy iterative quantum phase estimation algorithm using a single ancillary qubit: A two-qubit benchmark, Phys. Rev. A **76**, 030306(R) (2007).

[37] R. B. Griffiths and C.-S. Niu, Semiclassical fourier transform for quantum computation, Phys. Rev. Lett. **76**, 3228 (1996).

[38] R. D. Somma, Quantum eigenvalue estimation via time series analysis, New J. Phys. **21**, 123025 (2019).

[39] T. E. O'Brien, B. Tarasinski, and B. M. Terhal, Quantum phase estimation of multiple eigenvalues for small-scale (noisy) experiments, New J. Phys. **21**, 023022 (2019).

[40] K. Wan, M. Berta, and E. T. Campbell, Randomized quantum algorithm for statistical phase estimation, Phys. Rev. Lett. **129**, 030503 (2022).

[41] A. Szabo and N. S. Ostlund, *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory* (Dover, Mineola, New York, 1996).

[42] S. Bravyi, J. M. Gambetta, A. Mezzacapo, and K. Temme, Tapering off qubits to simulate fermionic Hamiltonians, arXiv:1701.08213 [quant-ph].

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

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

[45] B. Eastin and E. Knill, Restrictions on transversal encoded quantum gate sets, Phys. Rev. Lett. **102**, 110502 (2009).

[46] C. M. Dawson and M. A. Nielsen, The Solovay-Kitaev algorithm, Quantum Inf. Comput. **6**, 81 (2006).

[47] V. Kliuchnikov, K. Lauter, R. Minko, A. Paetznick, and C. Petit, Shorter quantum circuits, Quantum **7**, 1208 (2023).

[48] Riverlane, Supplementary material for "compilation of a simple chemistry application to quantum error correction primitives", https://github.com/riverlane/h2_compilation.

[49] H. Bombín, C. Dawson, R. V. Mishmash, N. Nickerson, F. Pastawski, and S. Roberts, Logical blocks for fault-tolerant topological quantum computation, PRX Quantum **4**, 020303 (2023).

[50] S. Bravyi and J. Haah, Magic-state distillation with low overhead, Phys. Rev. A **86**, 052329 (2012).

[51] C. Jones, Multilevel distillation of magic states for quantum computing, Phys. Rev. A **87**, 042305 (2013).

[52] A. G. Fowler, S. J. Devitt, and C. Jones, Surface code implementation of block code state distillation, Sci. Rep. **3**, 1939 (2013).

[53] J. Haah and M. B. Hastings, Codes and Protocols for Distilling T, controlled-S, and Toffoli Gates, Quantum **2**, 71 (2018).

[54] V. Graves, C. Sünderhauf, N. S. Blunt, R. Izsák, and M. Szőri, The electronic structure of the hydrogen molecule: A tutorial exercise in classical and quantum computation, arXiv:2307.04290 [physics.chem-ph].

[55] G. H. Low and I. L. Chuang, Hamiltonian simulation by qubitization, Quantum **3**, 163 (2019).

[56] G. H. Low and I. L. Chuang, Optimal Hamiltonian simulation by quantum signal processing, Phys. Rev. Lett. **118**, 010501 (2017).

[57] A. Gilyén, Y. Su, G. H. Low, and N. Wiebe, Quantum singular value transformation and beyond: Exponential improvements for quantum matrix arithmetics, in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019 (Association for Computing Machinery, New York, NY, USA, 2019), pp. 193–204.

[58] J. M. Martyn, Z. M. Rossi, A. K. Tan, and I. L. Chuang, Grand unification of quantum algorithms, PRX Quantum **2**, 040203 (2021).