Low-depth fermion routing without ancillas

Nathan Constantinides*,[1,2] Jeffery Yu*,[1,2] Dhruv Devulapalli,[1,2] Ali Fahimniya,[1,2] Luke Schaeffer,[1,3] Andrew M. Childs,[1,4] Michael J. Gullans,[1,4] Alexander Schuckert†,[1,2] and Alexey V. Gorshkov†[1,2]

[1] *Joint Center for Quantum Information and Computer Science, NIST/University of Maryland, College Park, Maryland 20742, USA*
[2] *Joint Quantum Institute, NIST/University of Maryland, College Park, Maryland 20742, USA*
[3] *Institute for Quantum Computing, University of Waterloo*
[4] *Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, Maryland 20742, USA*

(Dated: October 29, 2025)

Routing is the task of permuting qubits in such a way that quantum operations can be parallelized maximally, given constraints on the hardware geometry. When simulating fermions in the Jordan-Wigner encoding with qubits, a one-dimensional nearest-neighbor-connected geometry is effectively imposed on the system, independently of the underlying hardware, which means that naively, an $O(N)$ depth routing overhead is incurred. Recently, Maskara et al. [arXiv:2509.08898] demonstrated that this routing overhead can be reduced to $O(\log N)$ by decomposing general fermion routing into $O(\log N)$ interleave permutations of depth $O(1)$, using $\Theta(N)$ ancillary qubits and employing measurements and feedforward. Here, we exhibit an alternative construction that achieves the same asymptotic performance. We also generalize the result in two ways. Firstly, we show that fermion routing can be performed in depth $O(\log^2 N)$ *without* ancillas, measurements, or feedforward. Secondly, we construct efficient mappings with $O(\log^2 N)$ depth between all product-preserving ternary tree fermionic encodings, thereby showing that fermion routing in any such encoding can be done efficiently. While these results assume all-to-all connectivity, they also imply upper bounds for fermion routing in devices with limited connectivity by multiplying the fermion routing depth by the worst-case qubit routing depth.

I. INTRODUCTION

Standard quantum computers based on qubit architectures suffer from locality constraints. These constraints determine which pairs of qubits are allowed to interact, and which interactions may be implemented in parallel. Hence, these constraints present a challenge for performing quantum computation, and must be accounted for in order to implement quantum information processing tasks. One way to handle locality constraints is by performing routing, which is the task of implementing permutations of qubits. By permuting qubits, one may simulate all-to-all connectivity and therefore implement arbitrary quantum circuits with an overhead in circuit depth. In order to minimize this overhead, the problem of routing efficiently under interaction constraints has been studied using swap gates [1–3], teleportation and mid-circuit measurements [4–6], and dynamic reconfigurability [7, 8]. In particular, the worst-case depth of routing of qubits arranged in a one-dimensional line using reconfigurable neutral-atom hardware has been shown to be $O(\log N)$ [8, 9].

When simulating fermionic systems on qubit-based quantum computers, routing arises naturally due to the structure of fermion-to-qubit mappings. In the Jordan-Wigner encoding, each fermionic mode is represented by a qubit. To reproduce fermionic statistics, the qubits are arranged on a one-dimensional line, independent of the actual geometry of the Hamiltonian: when two fermions are exchanged between neighboring lattice sites on the Jordan-Wigner line, the wavefunction is given a minus sign. This effectively imposes a one-dimensional nearest-neighbor connectivity constraint, regardless of the underlying hardware geometry. To account for this constraint, fermion routing is typically performed using fermi-SWAP networks [10], in which layers of fSWAP=SWAP·CZ gates permute fermionic modes, where the CZ gate applies the minus sign when two modes are occupied by a fermion. Because of this one-dimensional nearest-neighbour structure, a general routing task on N modes uses $O(N)$ layers of such gates. While 1D $O(\log N)$-depth qubit routing schemes can in principle be directly used to decompose a general fermion permutation into $O(\log N)$ steps consisting of interleaves (riffle shuffles) [9, 11] or staircase permutations (in-order swaps) [8], this does not directly improve the overall asymptotic scaling due to the $O(N^2)$ CZ gates which need to be applied in each

* These authors contributed equally to this work.
† These authors contributed equally to the supervision of this work. gorshkov@umd.edu, schuckertalexander@gmail.com

interleave or staircase, which naively require $O(N)$ depth. Recently, Maskara et al. [11] showed that the interleave CZ circuit can in fact be compressed to depth $O(1)$ by using $\Theta(N)$ ancillas and measurement and feedforward, implying that an arbitrary permutation can be implemented in depth $O(\log N)$.

In this work, we show that staircase permutations can be compressed to depth $O(\log N)$ without ancillas, measurement, or feedforward.[†] This means that without ancillas, we achieve depth $O(\log^2 N)$ for arbitrary fermion routing. In addition, we generalize this result to arbitrary product-preserving ternary-tree encodings by proving that there exists a depth $O(\log^2 N)$ mapping between these encodings. Finally, we discuss consequences for simulation of fermionic time evolution.

More specifically, we show the following theorem:

Theorem 1. *For any product-preserving ternary tree encoding of fermions into N qubits, there exists a circuit of depth $O(\log^2 N)$ that implements any given fermionic permutation of N fermionic modes.*

We show this theorem by explicitly constructing the circuit, also enabling a practical implementation. In addition, we show the following corollary, using Ref. [11]:

Corollary 1. *For any product-preserving ternary tree encoding of fermions into N qubits, there exists a circuit of depth $O(\log N)$ with $O(N)$ ancillas and measurement and feedforward that implements any given fermionic permutation of N fermionic modes.*

The remainder of this manuscript is organized as follows. In Section II, we introduce definitions and notation. In Section III, we present a $O(\log^2 N)$-depth circuit using staircase permutations that performs any fermionic permutation under the Jordan-Wigner mapping. In Section IV, we present a $O(\log^2 N)$-depth circuit that converts between any two product-preserving ternary tree mappings. In Section V, we discuss applications to quantum simulation of fermionic models. In Section VI, we briefly discuss some directions for future work.

II. PRELIMINARIES

In this section, we introduce definitions and notation that are used throughout the paper.

We work with fermions in second-quantized form, where states represent the occupation of a given mode. Given N_f ordered fermionic modes, we denote a Fock state as $\left|x_0, \ldots, x_{N_f-1}\right\rangle_f$, where $x_k \in \{0, 1\}$ is the number of fermions occupying the kth mode. The subscript f distinguishes a Fock state from a qubit state, which has no subscript.

We denote by $\mathcal{H}_f$ the Hilbert space spanned by the Fock states and $\mathcal{H}_q$ the standard Hilbert space on N_q qubits.

Definition 1 (Fermion-to-qubit map)**.** A fermion-to-qubit mapping is an isometry $\phi\colon \mathcal{H}_f \to \mathcal{H}_q$.

Operators in $\mathcal{H}_f$ can be expressed by fermionic annihilation and creation operators a_k and $a_k^\dagger$, respectively. These are defined on Fock states as

$$a_k \left|x_0, \ldots, x_k = 0, \ldots, x_{N_f-1}\right\rangle_f = 0, \tag{1a}$$

$$a_k \left|x_0, \ldots, x_k = 1, \ldots, x_{N_f-1}\right\rangle_f = (-1)^{\sum_{j=0}^{k-1} x_j} \left|x_0, \ldots, x_k = 0, \ldots, x_{N_f-1}\right\rangle_f, \tag{1b}$$

$$a_k^\dagger \left|x_0, \ldots, x_k = 0, \ldots, x_{N_f-1}\right\rangle_f = (-1)^{\sum_{j=0}^{k-1} x_j} \left|x_0, \ldots, x_k = 1, \ldots, x_{N_f-1}\right\rangle_f, \tag{1c}$$

$$a_k^\dagger \left|x_0, \ldots, x_k = 1, \ldots, x_{N_f-1}\right\rangle_f = 0, \tag{1d}$$

and satisfy the anticommutation relations

$$\{a_j, a_k\} = \{a_j^\dagger, a_k^\dagger\} = 0, \qquad \{a_j, a_k^\dagger\} = \delta_{jk}. \tag{2}$$

[†] By using the $O(\log N)$-depth compression of CNOT ladders from Ref. [12], the interleave permutations in Ref. [11] can also be implemented without ancillas, measurement, or feedforward in $O(\log N)$ depth [13].

The set $\{a_k, a_k^\dagger\}_{k=0}^{N_f-1}$ spans the algebra of fermionic operators $\mathcal{B}(\mathcal{H}_f)$. Qubit operators can be expressed in terms of the Pauli basis $\{I_k, X_k, Y_k, Z_k\}_{k=0}^{N_q-1}$.

Given a fermion-to-qubit mapping ϕ, each fermionic operator $\mathcal{O}_f \in \mathcal{B}(\mathcal{H}_f)$ has a unique corresponding qubit operator $\mathcal{O}_q \in \mathcal{B}(\operatorname{im}\phi)$ satisfying

$$\mathcal{O}_q \phi(|\psi\rangle_f) = \phi\Big(\mathcal{O}_f |\psi\rangle_f\Big) \tag{3}$$

for every $|\psi\rangle_f \in \mathcal{H}_f$. That is, $\mathcal{O}_q = \phi \circ \mathcal{O}_f \circ \phi^{-1}$ acts on the mapped qubit states in the same manner that $\mathcal{O}_f$ acts on fermionic states. We denote by $\Phi\colon \mathcal{B}(\mathcal{H}_f) \to \mathcal{B}(\operatorname{im}\phi)$ the operator mapping $\mathcal{O}_f \mapsto \mathcal{O}_q$ induced by ϕ. Throughout this paper, we consider the case $N_f = N_q = N$. Then every fermion-to-qubit mapping ϕ is surjective, i.e., $\operatorname{im}\phi \cong \mathcal{H}_q$, and ϕ is a unitary map. We note that in the case $N_q > N_f$, superfast encodings are known which simulate Hamiltonians with bounded degree interaction graphs with $O(1)$ overhead. Namely, the Bravyi-Kitaev superfast encoding [14] satisfies this property with N_q scaling with the degree of the interaction graph d, $N_q = \Omega(dN_f)$.

For example, the classic Jordan-Wigner mapping [15], ϕ_{JW}, is given by simply treating the list of occupation numbers as a computational basis state in the qubit space:

$$\phi_{\mathrm{JW}}\big(|x_0, x_1, \ldots, x_{N-1}\rangle_f\big) = |x_0, x_1, \ldots, x_{N-1}\rangle\,. \tag{4}$$

Equation (4) induces a mapping of creation and annihilation operators to qubit operators as follows:

$$\Phi_{\mathrm{JW}}(a_k) = \frac{1}{2} Z_0 \otimes Z_1 \otimes Z_2 \otimes \cdots \otimes Z_{k-1} \otimes (X_k + iY_k), \tag{5a}$$

$$\Phi_{\mathrm{JW}}(a_k^\dagger) = \frac{1}{2} Z_0 \otimes Z_1 \otimes Z_2 \otimes \cdots \otimes Z_{k-1} \otimes (X_k - iY_k). \tag{5b}$$

Note that $\sigma_k^+ = \frac{1}{2}(X_k + iY_k)$ and $\sigma_k^- = \frac{1}{2}(X_k - iY_k)$ are simply the qubit lowering and raising operators, respectively, while the leading string of Pauli Zs implements the $(-1)^{\sum_{j=0}^{k-1} x_j}$ phase appearing in Eq. (1).

The Jordan-Wigner mapping is often considered the simplest fermion-to-qubit mapping due to its simplicity in mapping the states. However, its disadvantage is in the linear weight of the Pauli strings of mapped fermionic operators [16]. Consequently, many other mappings (see e.g. Refs. [14, 17–20]) have been developed. For example, the Bravyi-Kitaev transformation [14] maps individual fermion operators to log-weight Pauli strings. However, as argued in Section I, this should not be the sole figure of merit for a fermion-to-qubit map. Rather, the overall simulation cost depends on the extent to which multiple Hamiltonian terms may be implemented in parallel. Therefore, in this work, we focus on the task of implementing fermion permutations.

Definition 2 (Fermion permutation)**.** Let $\sigma\colon [N] \to [N]$ be a permutation. The fermion routing operator implementing σ is the operator U_σ satisfying

$$U_\sigma a_k U_\sigma^\dagger = a_{\sigma(k)} \quad \forall\, k \in [N] \tag{6a}$$

$$U_\sigma\, |0\ldots0\rangle_f = |0\ldots0\rangle_f\,. \tag{6b}$$

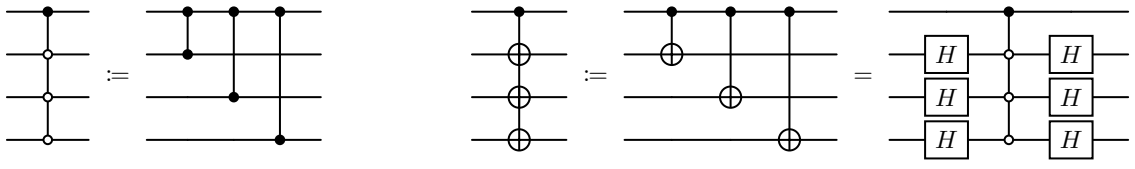
Note that condition (6a) uniquely specifies the action of U up to a phase, while (6b) fixes the phase. In particular, we can calculate the action of U on a general Fock state by writing

$$|x_0, \ldots, x_{N-1}\rangle_f = (a_0^\dagger)^{x_0} \cdots (a_{N-1}^\dagger)^{x_{N-1}}\, |0\ldots0\rangle_f \tag{7}$$

and letting U_σ conjugate each creation operator.

Central to our qubit circuits will be controlled-Z gates. We let $\mathrm{CZ}_{a,b}$ denote such a gate acting on qubits a and b. The effect of this is a phase $(-1)^{x_a x_b}$ on the qubit state $|x_a x_b\rangle$. When multiple CZ gates share a common qubit, we call this a CZ-fanout, as shown in Fig. 1(a).

CZ gates are related to CNOT gates by conjugating the target qubits with Hadamard gates, as depicted in Fig. 1(b). Another related gate is the parity gate, also known as the MOD_2 gate, where several CNOT gates share the same target qubit. The MOD_2 gate is related to the CNOT-fanout by conjugating all qubits with Hadamard gates [21].



(a) (b)

FIG. 1. (a) The CZ-fanout gate (left) is defined as a circuit consisting of one CZ gate between the control qubit (filled circle) and each target qubit (open circles). (b) The ordinary (CNOT) fanout is related to the CZ-fanout through conjugating each target qubit by Hadamards.

Another relevant unitary we will use is the parity transform P which acts on computational basis states as

$$P\left|x_0, x_1, \ldots, x_{N-1}\right\rangle = \left|p_0, p_1, \ldots, p_{N-1}\right\rangle, \text{ where } p_k = \bigoplus_{i=0}^{k} x_k. \tag{8}$$

To avoid confusion between the parity *gate* and the parity *transform*, we will henceforth refer to the former as the MOD_2 gate. It is well-known that all four of CZ-fanout, CNOT-fanout, MOD_2, and P can be implemented in a circuit with depth $O(\log n)$ [12, 21, 22]. We give another derivation of a log-depth circuit for P from a ternary tree perspective in Section IV.

III. PERMUTATIONS IN JORDAN-WIGNER

In this section, we give an explicit circuit construction for any fermionic permutation under the Jordan-Wigner mapping. We begin by building circuits for transpositions that swap two fermionic modes. A general permutation can then be implemented as a product of swaps.

Definition 3 (Fermionic swap)**.** The *fermionic swap* operator fSWAP_{jk} is the fermion permutation operator in Definition 2 that implements the transposition swapping modes j and k.

In the Fock basis, the fermionic swap operator acts on adjacent modes by swapping the occupation numbers and introducing a -1 phase if both are occupied [14]:

$$\mathrm{fSWAP}_{i,i+1}\left|x_1, x_2, \ldots, x_i, x_{i+1}, \ldots, x_N\right\rangle_f = (-1)^{x_i x_{i+1}}\left|x_1, x_2, \ldots, x_{i+1}, x_i, \ldots, x_N\right\rangle_f. \tag{9}$$

From this, the action of fSWAP_{ij} between modes with $i < j$ may be derived:

$$\mathrm{fSWAP}_{ij}\left|x_0, \ldots, x_i, \ldots, x_j, \ldots x_{N-1}\right\rangle_f = (-1)^p\left|x_0, \ldots, x_j, \ldots, x_i, \ldots, x_{N-1}\right\rangle_f, \tag{10}$$

with

$$p = x_i \sum_{k=i+1}^{j-1} x_k + x_j \sum_{k=i}^{j-1} x_k. \tag{11}$$

This equation also holds for the corresponding qubit states in the Jordan-Wigner encoding, which translates into two CZ-fanout gates followed by a qubit swap, as shown in Fig. 2.

Theorem 2. *Given any permutation $\sigma\colon [N] \to [N]$, there exists a circuit of depth $O(\log^2 N)$ that implements $\Phi_{\mathrm{JW}}(U_\sigma)$.*

At a high level, we proceed in two stages. First, we decompose an arbitrary permutation into a product of $\lceil \log_2 N \rceil$ layers of disjoint *staircase* permutations, which we define below. Then, we show how to implement each staircase permutation in depth $O(\log N)$, yielding a depth-$O(\log^2 N)$ routing circuit for any permutation.

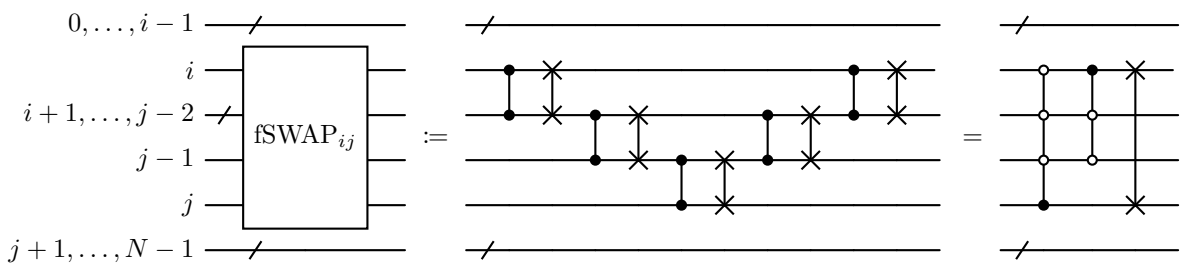


FIG. 2. The circuit for performing the swap of modes i and j in the Jordan-Wigner encoding. The two CZ-fanouts in the final equality are generated by commuting each CZ gate to the front of the SWAP circuit, and then collapsing the SWAP operators into a single SWAP of modes i and j.

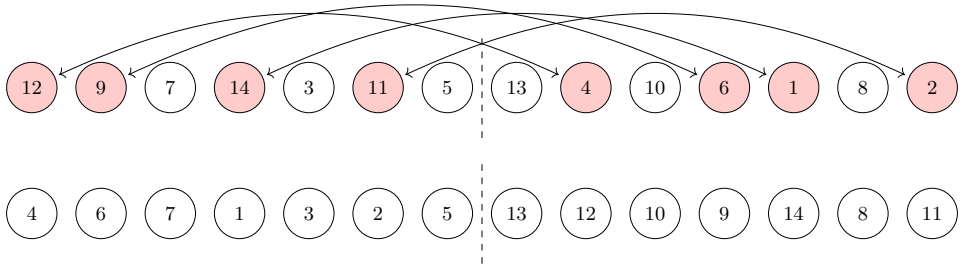


FIG. 3. In this example, each element (indicated by a circle) is labeled with its target position under a permutation σ. The first step of implementing this permutation using staircase permutations is shown. Red marked elements have destinations across the dashed midpoint of the array, and are arranged in pairs and swapped in one staircase permutation step. Following this staircase step, the algorithm is run recursively on each side of the midpoint, treating the left and right halves separately.

We remark that these two stages resemble the construction in Ref. [11], where an arbitrary permutation is decomposed into $O(\log N)$ layers of special efficient permutations. However, we differ in the choice of these special efficient permutations: we use ones that we call staircase permutations, while Ref. [11] uses others that they call interleaves. While some permutations are both interleaves and staircase permutations, neither is a subset of the other. Every staircase permutation can be implemented as a product of two interleaves [8, Sec. SI], and it remains open whether every interleave can be implemented via a constant number of staircase permutations.

Lemma 1. *Let a* staircase *permutation be any permutation that can be written as a sequence of disjoint transpositions,* $(m_1, n_1), \ldots, (m_k, n_k)$*, such that the sequences* $m_1, m_2, \ldots, m_k$ *and* $n_1, n_2, \ldots n_k$ *are strictly increasing, and* $m_k < n_1$*. We call the interval* $[m_1, n_k]$ *its* range*. Then every permutation* $\sigma \colon [N] \to [N]$ *can be written as a composition of* $\lceil \log_2 N \rceil$ *layers of staircase permutations, where each layer consists of a product of staircase permutations with disjoint ranges.*

Proof. In the first layer, we partition $[N]$ into two halves: $L = \{1, 2, \ldots, \lfloor N/2 \rfloor\}$ and $R = \{\lfloor N/2 \rfloor + 1, \ldots, N\}$. Let $x_1 < x_2 < \cdots < x_k$ be all the elements of L such that $\sigma(x_i) \in R$ for every x_i. Then there must be exactly k elements of R, which we label $y_1 < y_2 < \cdots < y_k$, with $\sigma(y_j) \in L$ for every y_j. We apply the staircase permutation $\sigma_1 = (x_1, y_1) \cdots (x_k, y_k)$, as depicted in Fig. 3. After applying σ_1, the left half contains exactly the elements mapping to L and the right half contains exactly the elements mapping to R (i.e. $(\sigma \circ \sigma_1^{-1})(L) = L$ and $(\sigma \circ \sigma_1^{-1})(R) = R$).

In subsequent layers, we recursively apply this construction within $\sigma_1(L)$ and $\sigma_1(R)$. Since the halves are disjoint, the two resulting staircase permutations are disjoint. As each layer operates on subsets of half the size of the prior layer, there are $\lceil \log_2 N \rceil$ layers. □

Next, we introduce a technical circuit construction that is equivalent to [11, Lemma 3] but that we rederive

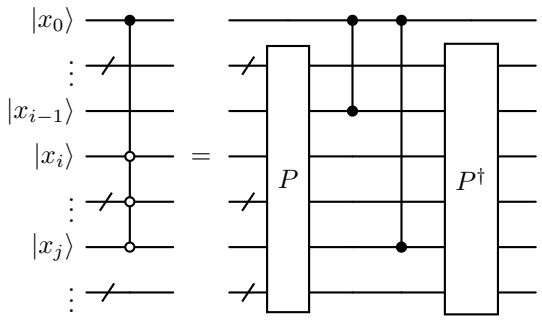


FIG. 4. The effect of conjugating the targets of a CZ-fanout with the parity (P) gate, as defined in Eq. (8). The fanout, which has control 0 and targets $\{i, \ldots, j\}$, becomes two CZ gates, one $\mathrm{CZ}_{0,i-1}$ and another $\mathrm{CZ}_{0,j}$. If the first qubit in P is i, then the $\mathrm{CZ}_{0,i-1}$ gate is not present.

here to demonstrate ancilla-free techniques.

Lemma 2. *Let a* contiguous-range CZ-fanout circuit *be a circuit consisting of* CZ*-fanout gates with the following two properties:*

1. *Each fanout has a distinct control qubit on which no other gates act, and*
2. *Each fanout has targets spanning a contiguous range of qubits.*

Then any contiguous-range CZ*-fanout circuit with support on n qubits can be implemented in $O(\log n)$ depth.*

Proof. As shown in Fig. 4, a contiguous-range CZ-fanout whose targets are conjugated with the parity (P) transformation turns into at most two CZ gates. Specifically, if the CZ-fanout originally has control 0 and targets $\{i, i+1, \ldots, j\}$, it becomes the gates $\mathrm{CZ}_{0,j}$ and $\mathrm{CZ}_{0,i-1}$ under conjugation by P, unless i is the first qubit acted on by P, in which case it becomes the single $\mathrm{CZ}_{0,j}$ gate. Using this insight, we begin by conjugating by P the shared target qubits of all k contiguous-range CZ-fanouts, as shown in the example of Fig. 5. As the control lines of each CZ-fanout are distinct, we can separate the resulting CZ gates into two sets, marked blue and red, so that within each set, each CZ acts on a distinct qubit from the original set of controls. We depict this in Fig. 6. Finally, using the property that all CZ gates commute, we observe that the blue CZ gates and red CZ gates may be collapsed into two layers of at most k CZ-fanouts, where in each layer each fanout acts on disjoint qubits. Since a CZ-fanout can be done in depth $O(\log n)$, this completes the proof. □

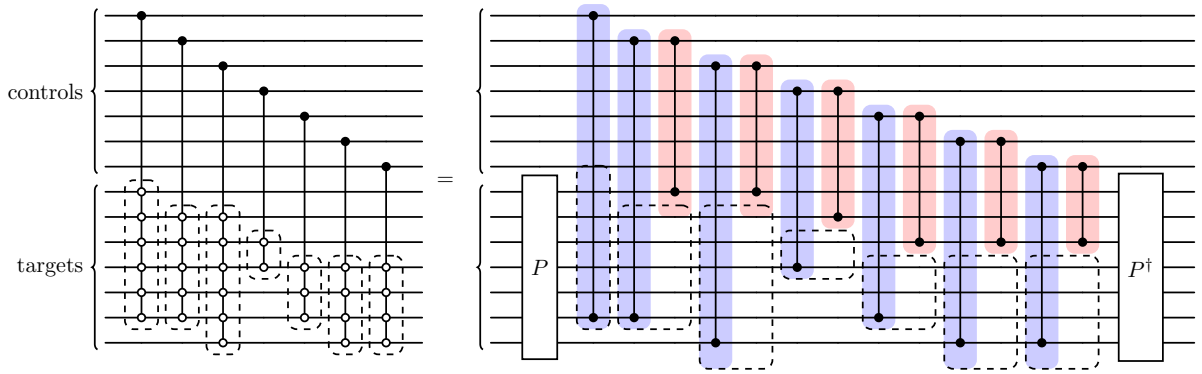


FIG. 5. An example of a contiguous-range CZ-fanout circuit. When transformed into the parity basis, the CZ-fanouts turn into a sequence of individual CZ gates.

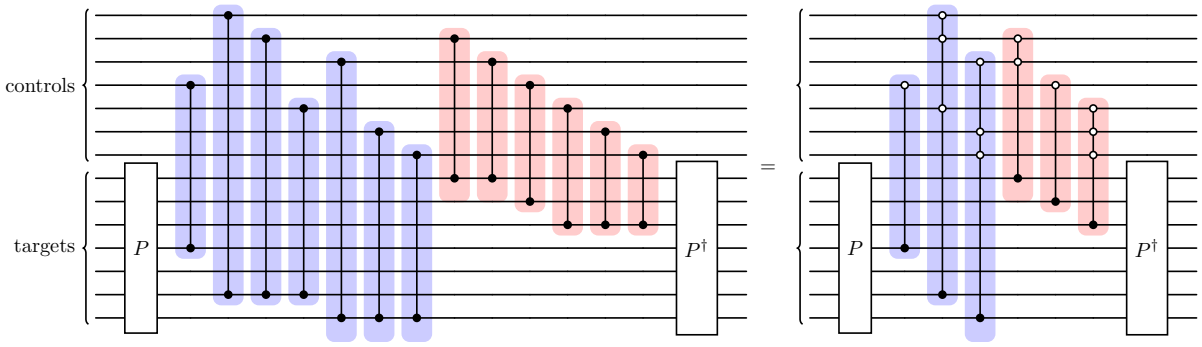


FIG. 6. Continuing the example from Fig. 5, the CZ gates can be grouped into two sets (highlighted blue and red) of CZ-fanouts. Within each set, the CZ-fanouts act on (i.e., have as their targets) disjoint sets of qubits.

We are now ready to prove Theorem 2.

Proof of Theorem 2. We begin by decomposing σ into $O(\log N)$ layers of disjoint staircase permutations using Lemma 1. Since, in each layer, each staircase permutation has a disjoint range of elements, by promoting each of the transpositions in the $\lceil \log_2 N \rceil$ layers of staircase permutations into fermionic swaps in the Jordan-Wigner encoding, one obtains a fermion routing circuit similarly made of $\lceil \log_2 N \rceil$ layers of fermionic staircase permutations acting on disjoint sets of qubits. This is seen by the property that each fSWAP_{ij} acts only on qubits $\{i, i+1, \ldots, j\}$, so each fermionic staircase permutation acts on disjoint sets of qubits in each layer, and thus the problem of fermion routing is reduced to implementing arbitrary fermionic staircase permutations, which we now show can be done in $O(\log N)$ depth.

To implement a general fermionic staircase permutation $(m_1, n_1) \ldots (m_k, n_k)$, we begin with a naive circuit implementing each transposition as two CZ-fanouts followed by a qubit SWAP, as shown in an example in Fig. 7. We follow this example throughout the rest of our proof, though our discussion is general to any staircase permutation.

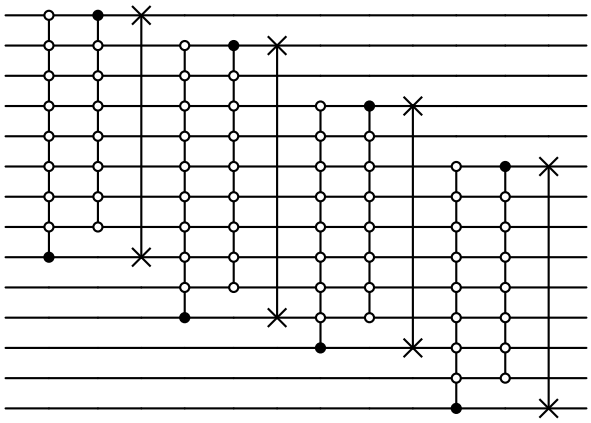


FIG. 7. Example of a staircase of fSWAPs expanded in terms of CZ-fanouts and SWAP gates (see Fig. 2).

Note that all CZ gates commute with each other, while commuting a SWAP with a CZ simply changes which qubits the CZ acts on, transporting the target qubit of a CZ from one end of the SWAP to the other. We commute all SWAP gates to the right of the naive circuit, shown for our example in Fig. 8. This updates the CZ-fanouts by reassigning their targets, while the staircase property of the permutation ensures that the controls of each fanout do not change. Furthermore, we commute the CZ-fanouts controlled by a qubit in $\{n_i\}_i$ to the left and group them as circuit (1), and then group the fanouts with a control qubit in $\{m_i\}_i$

as circuit (2). These are depicted for the example in dashed red boxes in Fig. 8. To elucidate a pattern generated circuit, we mark those qubits in the set $\{m_i\}_i$ blue, and those in the set $\{n_i\}_i$ red. The remaining qubits are unmarked and labeled in order starting from 1.

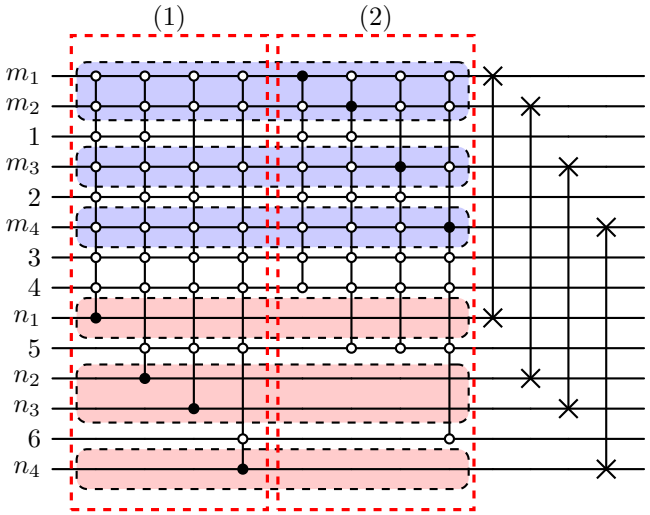


FIG. 8. The example of a staircase of fSWAPs continued from Fig. 7. The properties of SWAP gates and commutation of CZ gates are used to bring each CZ-fanout to the left of the circuit. We commute the CZ-fanouts into groups (1) and (2). In group (1), each fanout has a control qubit in $\{n_i\}_i$ and in group (2) each fanout has a control qubit in $\{m_i\}_i$. Qubits $\{m_i\}_i$ and $\{n_i\}_i$ are labeled and marked blue and red, respectively. The rest of the qubits are labeled starting from 1 in order, and unmarked.

We make some observations on the structure of the fanouts in the commuted circuit, such as in the example in Fig. 8, starting with circuit (1). Specifically, the first fanout of each fSWAP_{m_i,n_i} operation in the naive circuit (e.g. Fig. 7) has control n_i, targets $\{m_i, m_i+2, \ldots, n_i-1\}$, and is commuted through the set of SWAP gates $\{\text{SWAP}_{m_1,n_1}, \text{SWAP}_{m_2,n_2}, \ldots, \text{SWAP}_{m_{i-1},n_{i-1}}\}$. This has the effect of reassigning a subset of its targets $\{n_1, n_2, \ldots, n_{i-1}\}$ to be those qubits $\{m_1, m_2, \ldots, m_{i-1}\}$. Thus, its targets will always be given by the set of qubits $\{m_j\}_j$, marked in blue, and the unmarked qubits between m_i and n_i, $\{m_i+1, m_i+2, \ldots, n_i-1\} \setminus (\{m_j\}_j \cup \{n_j\}_j)$, as is seen in Fig. 8. Similarly, in circuit (2), the fanout with control m_i originates from the second fanout of the fSWAP_{m_i,n_i} operation in the naive circuit (e.g. Fig. 7) with control m_i and targets $\{m_i+1, m_i+2, \ldots, n_i-1\}$. It is commuted through the set of SWAP gates $\{\text{SWAP}_{m_1,n_1}, \text{SWAP}_{m_2,n_2}, \ldots, \text{SWAP}_{m_{i-1},n_{i-1}}\}$, reassigning the subset of its targets $\{n_1, n_2, \ldots, n_{i-1}\}$ to the qubits $\{m_1, m_2, \ldots, m_{i-1}\}$. Thus, its targets in circuit (2) will be given by the set of blue qubits $\{m_j\}_j \setminus \{m_i\}$ and every unmarked qubit between m_i and n_i, $\{m_i+1, m_i+2, \ldots, n_i-1\} \setminus (\{m_j\}_j \cup \{n_j\}_j)$ (e.g. Fig. 8).

We next simplify each of the circuits (1) and (2), as shown in the examples of Figs. 9 and 10(a), respectively. In both circuits, we first reorder the qubits such that the blue qubits $\{m_i\}_i$ come first, then the red $\{n_i\}_i$, then the unmarked qubits. In circuit (1) (e.g. Fig. 9), we split each fanout gate into two fanout gates, one with control n_i and blue targets $\{m_j\}_j$, and the other with control n_i and unmarked targets qubits $\{m_i+1, m_i+2, \ldots, n_i-1\} \setminus (\{m_j\}_j \cup \{n_j\}_j)$. Lastly, in circuit (2) (e.g. Fig. 10(a)), we cancel the targets of each fanout with control m_i on the qubits $\{m_j\}_j \setminus \{m_i\}$ using the circuit identity shown in Fig. 10(b). This leaves only the unmarked qubit targets $\{m_i+1, m_i+2, \ldots, n_i-1\} \setminus (\{m_j\}_j \cup \{n_j\}_j)$. Notice that for any i, the sets of blue qubits $\{m_j\}_j$, and unmarked qubits between m_i and n_i, $\{m_i+1, m_i+2, \ldots n_i-1\} \setminus (\{m_j\}_j \cup \{n_j\}_j)$, now form contiguous ranges in the new ordering. Thus, each of the final circuits (1) and (2) (as in Fig. 9 and Fig. 10(a)) are composed of two and one contiguous-range CZ-fanout circuits respectively, which, as proven in Lemma 2, have depth $O(\log N)$. □

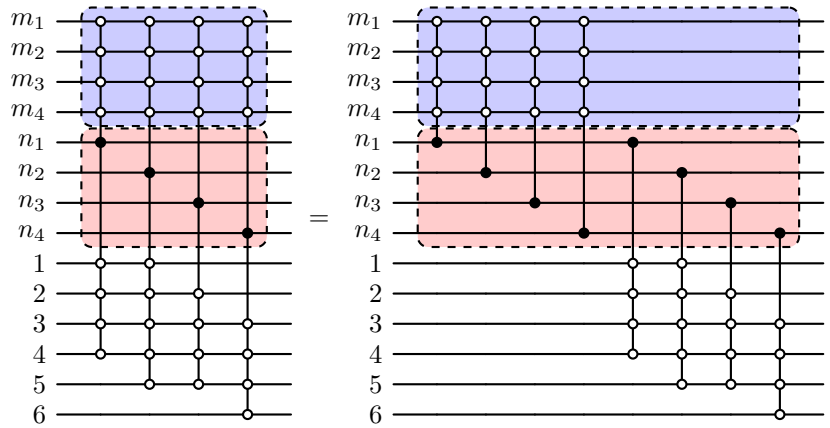


FIG. 9. Circuit (1) from Fig. 8 after reordering the qubits to place the qubits in $\{m_i\}_i$ and $\{n_i\}_i$ at the top of the circuit.

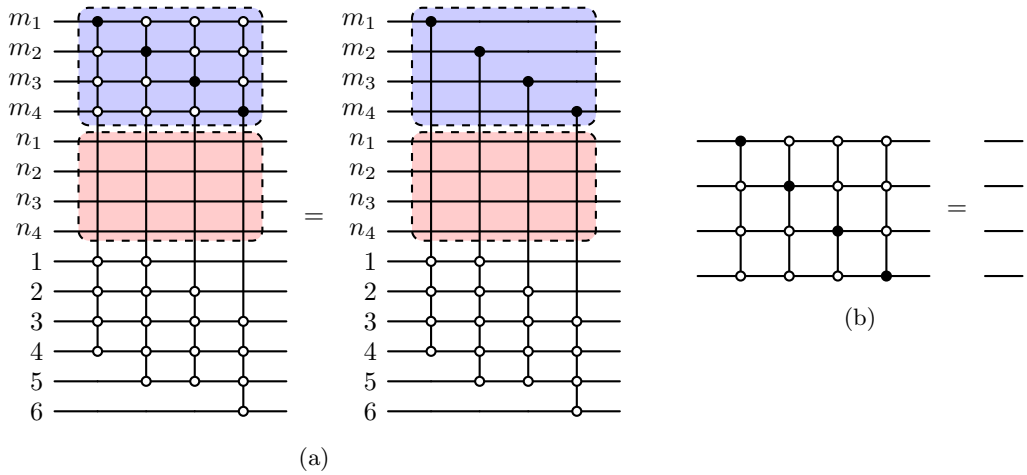


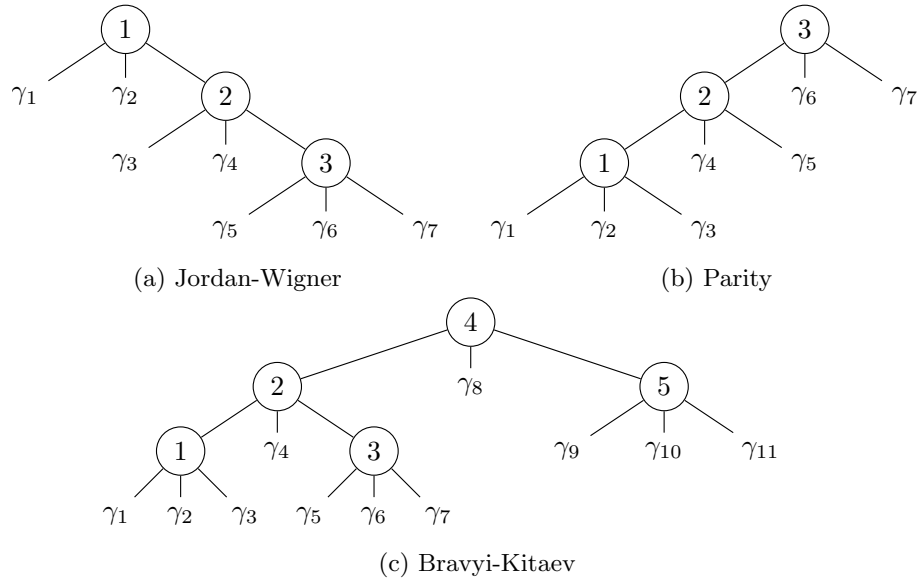
FIG. 10. (a) Circuit (2) of Fig. 8 following the same reordering of qubits as in Fig. 9. The identity in (b) is used to cancel each fanout target on the qubits $\{m_i\}_i$. (b) A sequence of k CZ-fanouts, in which each fanout has a unique control line and $k-1$ other shared targets reduces to the identity, as a CZ_{ij} gate is generated for every pair of qubits $i \neq j$.

IV. TRANSFORMATIONS BETWEEN ENCODINGS

In this section, we extend the regime of fermion-to-qubit mappings for which the fermion permutation circuit of Ref. [11] and Theorem 2 is applicable. We show that a broad range of mappings, called product-preserving ternary tree mappings, can be transformed into the Jordan-Wigner mapping in depth $O(\log^2 N)$. Consequently, routing in those mappings can also be implemented in depth $O(\log^2 N)$ by first transforming to Jordan-Wigner, applying our routing circuit, and then transforming back to the original map.

Let ϕ_1 and ϕ_2 be two fermion-to-qubit mappings over the same fermion and qubit spaces. Since we assume $\dim \mathcal{H}_f = \dim \mathcal{H}_q = 2^N$, they are unitary maps. Then there exists a unitary $U \in \mathcal{B}(\mathcal{H}_q)$ transforming between the two mappings such that $\phi_2 = U\phi_1$, namely $U = \phi_2\phi_1^{-1}$. This transformation acts on operators via conjugation:

$$\Phi_2(\mathcal{O}_f) = U\Phi_1(\mathcal{O}_f)U^\dagger. \tag{12}$$



(a) Jordan-Wigner (b) Parity

(c) Bravyi-Kitaev

FIG. 11. Many common fermion-to-qubit mappings, including (a) Jordan-Wigner, (b) Parity, and (c) Bravyi-Kitaev, can be expressed as ternary trees. All three examples shown are binary-shaped ternary tree mappings.

We construct efficient circuits implementing U when ϕ_1 and ϕ_2 lie in a class of fermion-to-qubit mappings known as product-preserving ternary tree mappings.

Ternary tree mappings, also known as qubit trees in other contexts [23], were introduced in Ref. [19] to give a mapping that provably minimizes the Pauli weight of $\Phi(a_k)$ averaged over all k. However, the initial construction only focused on the mapping of operators and not of states. Subsequent works [24, 25] extended the construction to study properties of the mapping of states. In this paper, we use a formulation of ternary trees adapted from Ref. [26]. See any standard classical algorithms text (e.g. Ref. [27]) for the basics of trees and tree traversal algorithms.

Definition 4. A *full ordered ternary tree* is a rooted tree where each node has exactly 0 or 3 children. A node with 0 children is a *leaf* and a node with 3 children is a *parent*. The 3 children of a given parent are ordered and named the left, middle, and right child.

Note that a full ternary tree with N parent nodes always has $2N + 1$ leaves.

Definition 5 (Ternary tree mapping)**.** A ternary tree mapping consists of the following information:

1. A full ordered ternary tree, which we call the *shape* of the ternary tree mapping.
2. A labeling of the N parents by $1, \ldots, N$. Each labeled parent corresponds to a qubit.
3. A labeling of the $2N + 1$ leaves by $\gamma_1, \ldots, \gamma_{2N+1}$. These label the Majorana operators.

Reference [19] showed that a ternary tree mapping with N parents gives a fermion-to-qubit mapping from N fermionic modes to N qubits by associating each leaf γ_j with a Pauli string as follows. Traverse down the tree from the root to γ_j. Along the path, taking the left, middle, or right child of qubit k appends to the Pauli string a Pauli X_k, Y_k, or Z_k, respectively. The mapping Φ of fermionic operators is then defined by $\Phi(a_k) = \gamma_{2k-1} + i\gamma_{2k}$. The last leaf γ_{2N+1} is redundant and discarded. Throughout this work, we do not explicitly keep track of the signs of the Majorana Paulis—as detailed in Ref [25], Pauli sign changes give an equivalent encoding up to single-qubit Paulis which take depth 1, and as such do not affect asymptotic scaling.

Definition 6 (Binary subtree)**.** Given a ternary tree, the binary subtree is the subgraph induced by the root and recursively all left and right children of nodes in the subtree. We call a tree *binary shaped* if all N qubit nodes lie in the binary subtree.

The Jordan-Wigner, Bravyi-Kitaev, and Parity (defined as $\phi_P |x_0, \ldots, x_{N-1}\rangle_f = |p_0, \ldots, p_{N-1}\rangle$ with p_k given by Eq. (8)) mappings can all be represented as binary-shaped ternary tree mappings [24], as shown in Fig. 11. Furthermore, in all three examples, the qubits are always labeled via an inorder traversal, and the leaves are labeled in order from left to right. (Recall that an inorder traversal is an ordering of nodes where we first recursively perform an inorder traversal of the left subtree, then visit the root, and then recursively perform an inorder traversal of the right subtree.)

To design circuits that transform between mappings, we first recall the effects of certain gates on ternary tree mappings.

Lemma 3 ([26, 28])**.** *Let ϕ be a ternary tree mapping. Then for each U below, $U\phi$ is another ternary tree mapping related to ϕ as described.*

(a) $U = \mathrm{SWAP}_{jk}$: Swap the qubit labels j and k in the tree.

(b) $U = \mathrm{CNOT}_{jk}$, where node j is the left child of node k: Perform the right tree rotation of qubit node j about node k in the binary subtree, preserving the parents of middle children. The right tree rotation is defined as shown in Fig. 12.

(c) $U = \mathrm{CNOT}_{jk}$, where k is the right child of node j: Perform the left tree rotation of qubit node k about node j in the binary subtree. This is the inverse of (b) and is defined as shown in Fig. 12.

(d) $U = S_k$, where $S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$ is the phase gate: Swap the left and middle children (carrying along their subtrees) of the node labeled k.

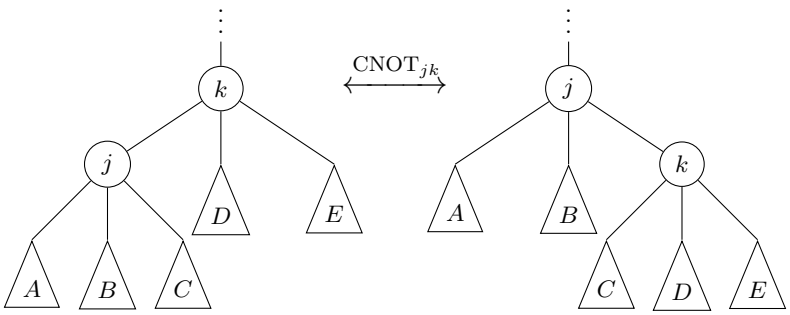


FIG. 12. Tree rotation corresponding to the adjoint action of the CNOT_{jk} transformation. The triangles can be leaves or contain further subtrees.

Theorem 3 (Folklore)**.** *There exists a* CNOT *circuit of depth $O(\log N)$ that transforms between the Jordan-Wigner, Bravyi-Kitaev, and Parity mappings.*

Proof. First, we demonstrate a transformation between Bravyi-Kitaev and Jordan-Wigner. We construct $O(\log N)$ layers of disjoint CNOT gates to transform the Bravyi-Kitaev tree to the Jordan-Wigner tree. Call the *right spine* of a tree the root and all nodes which are right descendants of nodes in the spine. In each layer, iterate through every node k on the right spine with a left child j and perform CNOT_{jk} to rotate node j onto the right spine. For each node not already on the right spine, its distance to the right spine decreases by 1 in each iteration. For Bravyi-Kitaev, the farthest starting node from the right spine has distance $O(\log N)$, so this procedure will terminate in $O(\log N)$ iterations. Since tree rotations leave invariant the inorder traversal, the resulting ternary tree also has qubits and leaves each labeled via an inorder traversal, thus exactly matching the Jordan-Wigner tree. An example of this protocol is illustrated in Fig. 13.

The transformation from Bravyi-Kitaev to Parity is simply the mirrored version of the above, with the left spine and left rotations instead. These transformations are unitary and invertible, so by composing Bravyi-Kitaev-to-Parity with the inverse of Bravyi-Kitaev-to-Jordan-Wigner, we obtain a log-depth circuit transforming between Jordan-Wigner and Parity. □

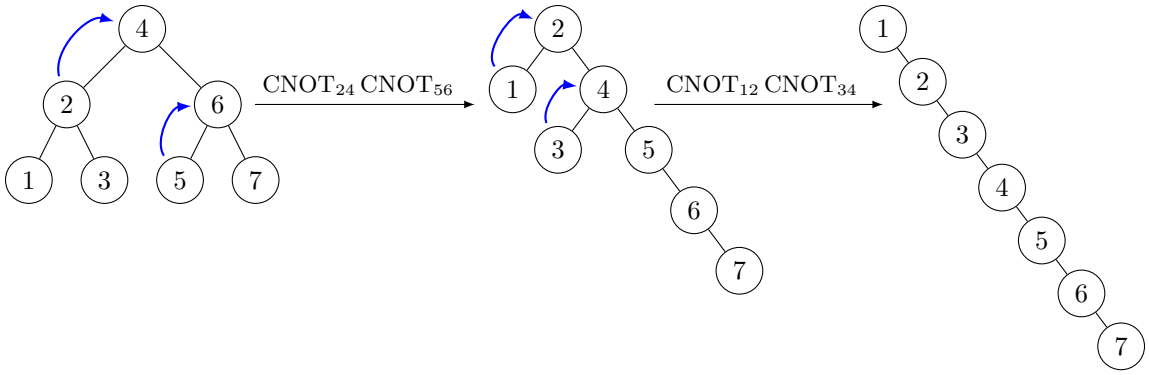


FIG. 13. For $N = 7$ qubits, a protocol transforming the Bravyi-Kitaev tree into the Jordan-Wigner tree using $\lfloor \log_2 N \rfloor$ layers of CNOT gates. The Majorana leaves are not drawn but are always implicitly labeled in order.

We remark that, while Theorem 3 was formulated in the context of fermion-to-qubit mappings, the circuits can also transform qubit states in the same manner, outside the context of any fermions. In particular, the same circuit constructed in Theorem 3 to transform Jordan-Wigner to Parity also gives a log-depth circuit implementing the parity transform [Eq. (8)] on qubit states. We note that this reproduces the same circuit as the well-known divide-and-conquer CNOT circuit for parity.

Algorithm 1 Circuit transforming any binary-shaped ternary tree into the Jordan-Wigner shape

```
while tree is not Jordan-Wigner shape do
    for all nodes k on the right spine do
        if k has a non-leaf left child j then
            Apply CNOT_jk
```

We also remark that the technique of rotating nodes onto the right spine is not restricted to starting from Bravyi-Kitaev. Rather, this technique can be applied to transform any binary-shaped ternary tree to the Jordan-Wigner shape and is detailed in Algorithm 1.

Lemma 4. *Let T be a binary-shaped ternary tree mapping and let d be the furthest distance from any node to the right spine. There exists a* CNOT *circuit of depth d that transforms between T and the Jordan-Wigner tree shape.*

Proof. The circuit is constructed by Algorithm 1. For each node not already on the right spine, its distance to the right spine decreases by 1 in each iteration of the **while** loop. Therefore the **while** loop runs d times. Within each iteration, all CNOT_{jk} gates can be applied in parallel as they act on disjoint qubits. Therefore the resulting circuit has depth d. □

Theorem 4. *There exists a* CNOT *circuit of depth $O(\log N)$ transforming between any two binary-shaped ternary trees on N qubits sharing the same ordering of Majorana leaf labeling.*

Proof. For this proof, we only keep track of the qubit nodes; the leaves are always attached in order since tree rotations leave the inorder traversal invariant.

It suffices to show that there exists a circuit of depth $O(\log N)$ transforming any binary-shaped tree to the Jordan-Wigner tree shape. To do so, we first use parallel tree rotations to transform an arbitrary binary tree into a binary tree with depth $O(\log N)$. The depth upper bounds the maximum distance of any node from the right spine, so by Lemma 4 we can transform the log-depth tree into the Jordan-Wigner shape in depth $O(\log N)$. By the inorder invariant of tree rotations, the Majorana leaves remain labeled in the same order. The only remaining degree of freedom is in the qubit labels, which can be fixed in depth two by decomposing the desired qubit permutation into two layers of SWAP gates [29].

To attain a log-depth binary-shaped tree, we introduce some notation to identify the imbalances in the tree. The *weight* of a node A, denoted $\mathrm{wt}(A)$, is the number of leaves in the subtree rooted at A. The *halving depth* of A, denoted $\mathrm{hd}(A)$, is the minimum number of levels below A such that every descendant B of A in that level has at most half the weight, i.e., $\mathrm{wt}(B) \leq \frac{1}{2}\,\mathrm{wt}(A)$.

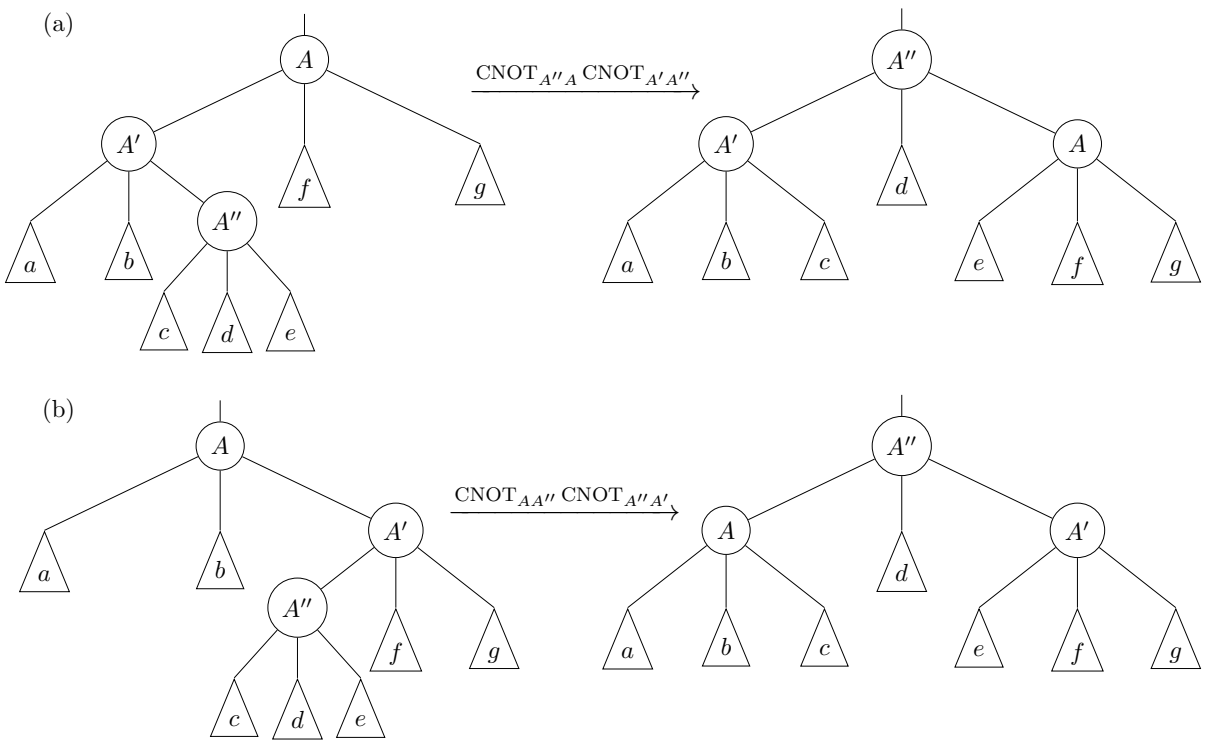


FIG. 14. In Theorem 4, depending on the shape of the heavy path, these rotations may be applied. In both cases, A''' can be the root of either subtree c, d or e.

There is clearly at most one node per level with weight $> \frac{1}{2}\,\mathrm{wt}(A)$—there is not enough weight in the whole subtree for more than one. In particular, some B exists $\mathrm{hd}(A)-1$ steps below A such that $\mathrm{wt}(B) > \frac{1}{2}\,\mathrm{wt}(A)$. By inclusion, all the ancestors of B up to A are also above the threshold. Inspired by this, we define the *heavy path from* A to be the sequence of descendants (of length $\mathrm{hd}(A)$) that have weight $> \frac{1}{2}\,\mathrm{wt}(A)$.

The algorithm runs $O(\log N)$ *rounds*. In each round, we consider all nodes where the depth is a multiple of 3, e.g., the root is depth 0, its great-grandchildren are depth 3, and so on. For each of these nodes A, if $\mathrm{hd}(A) \geq 4$ then the heavy path begins with nodes A, A', A'', A'''. The algorithm will shorten the heavy path using tree rotations, but the choice of rotation depends on which child A' is to A, and similarly on A'' relative to A'. Table I summarizes which rotation to use for each case. The key property is that in all cases, we reduce the length of three steps in a heavy path to two or less.

Now consider how a round of the algorithm affects the halving depth $d = \mathrm{hd}(X)$ of an arbitrary node X. The heavy path has d nodes, $X = X^{(1)}$ through $X^{(d)}$. The last 3 ($X^{(d-2)}$, $X^{(d-1)}$, $X^{(d)}$) are too close to the end to trigger a tree rotation in the algorithm. Of the remaining nodes, at least $\lfloor \frac{d}{3} \rfloor - 1$ appear on a level that is a multiple of three.

The tree rotations usually change the root of the subtree; call the new root Y. The final element of the original heavy path, $X^{(d)}$, exists in the subtree and still has weight $\mathrm{wt}(X^{(d)}) > \frac{1}{2}\,\mathrm{wt}(Y)$. However, the algorithm has applied no tree rotation that could affect $X^{(d)}$, so its children have weight $\leq \frac{1}{2}\,\mathrm{wt}(Y)$. Hence, the halving depth of Y is its distance from $X^{(d)}$. The distance between the subtree's root and $X^{(d)}$ decreases

		A' to A''	
		left	right
A to A'	left	CNOT$_{A''A'}$ (Fig. 12)	CNOT$_{A''A}$ CNOT$_{A'A''}$ (Fig. 14(a))
	right	CNOT$_{AA''}$ CNOT$_{A''A'}$ (Fig. 14(b))	CNOT$_{A'A''}$ (Fig. 12)

TABLE I. The choice of tree rotations in Algorithm 2.

by $\lfloor \frac{d}{3} \rfloor - 1 \geq \frac{d-5}{3}$ from the tree rotations, so

$$\mathrm{hd}(Y) \leq d - \frac{d-5}{3} \leq \frac{2d+5}{3}.$$

We conclude from this inequality that the halving depth of any subtree decreases geometrically to a constant. The initial halving depth of any subtree is at most the (standard) depth of the tree, and thus bounded by n. It follows that after $O(\log N)$ rounds, the halving depth of any subtree is $O(1)$.

Suppose the halving depth is bounded by a constant H for all subtrees. Then the (standard) depth of the tree is at most $H(\log_2(n) + O(1))$, since the weight is $2n+1$ at the root, halves every H levels, and never goes lower than 3. Since $H = O(1)$, we conclude that the standard depth is at most $O(\log N)$. □

Algorithm 2 Circuit transforming any binary-shaped ternary tree into the Jordan-Wigner shape

```
procedure BALANCE-BINARY
    while some node has hd ≥ 6 do
        for all nodes A with depth divisible by 3 do
            if hd(A) ≥ 4 then
                A′, A″, A‴ ← first three nodes in heavy path from A
                Apply CNOT rotations according to Table I
Apply BALANCE-BINARY to the starting tree
Apply Algorithm 1 on the resulting tree
```

Definition 7 (Product preservation)**.** A fermion-to-qubit mapping ϕ is product-preserving if $\phi(|\psi\rangle_f)$ is a computational basis state in $\mathcal{H}_q$ for every Fock basis state $|\psi\rangle_f$.

Theorem 5. *There exists a circuit of depth $O(\log^2 N)$, consisting of* CNOT *and S gates, transforming between any two product-preserving ternary tree mappings.*

Proof. It suffices to give an algorithm transforming any product-preserving ternary tree mapping into the Jordan-Wigner map. At a high level, we proceed in two stages. In the first stage, we apply Algorithm 3 to map any ternary tree into the Jordan-Wigner shape. In the second stage, we use the theory of product-preserving ternary trees from Ref. [25] to argue that the resulting tree cannot be too far from the actual Jordan-Wigner tree.

To begin the implementation of Algorithm 3, as illustrated in Fig. 15, we first apply Algorithm 2 to transform the binary subtree of the root, as well as the binary subtree of every middle child node, into one with no left children. All of these subtrees are independent of each other, so all their transformations can be done in parallel in depth $O(\log N)$.

Now we are left with a ternary tree where all left children are leaves. For every qubit node k that has a middle child qubit node, apply the phase gate S on qubit k. This swaps the left and middle children (Lemma 3) so that the left child has a qubit node and middle child a leaf. This makes the tree binary-shaped, so we can apply Algorithm 2 to convert it into the Jordan-Wigner tree shape, namely a single right spine. This completes the first stage.

For the second stage, we note that if $|\psi\rangle$ is a computational basis state, then so are $\mathrm{CNOT}_{jk}|\psi\rangle$ and $S_k|\psi\rangle$ for any j and k. Since all gates used in Algorithm 3 maintain the product preservation property, our resulting tree is also product-preserving. In particular, the vacuum state $|0,\ldots,0\rangle_f$ maps to some computational basis state $|q_1,\ldots,q_n\rangle$. We apply the Pauli gates $X_1^{q_1} \otimes \cdots \otimes X_n^{q_n}$ so that the vacuum state now maps to the computational basis state $|0^n\rangle$. This has no effect on the tree shape, since conjugation of a Pauli string by a Pauli can only multiply the string by a phase factor.

We now have a ternary tree with the Jordan-Wigner shape that maps the vacuum state to $|0^n\rangle$. By [25, Lemma 5.9], all ternary tree mappings with the Jordan-Wigner shape mapping the vacuum state to $|0^n\rangle$ must be equivalent to the exact Jordan-Wigner mapping up to a fermionic permutation, pair braiding (paired Majoranas $(\gamma_{2k-1}, \gamma_{2k})$ transforming to $(\gamma_{2k}, -\gamma_{2k-1})$), and Pauli sign changes. The latter two may be fixed by constant-depth single-qubit gates, leaving the fermion permutation as the only degree of freedom.

This means that, given any two product-preserving ternary-tree mappings ϕ_1 and ϕ_2, there exist circuits C_1 and C_2 of depth $O(\log^2 N)$ such that $C_1\phi_1 = \phi_{\mathrm{JW}}U_\sigma$ and $C_2\phi_2 = \phi_{\mathrm{JW}}U_\tau$ for some permutations $\sigma, \tau\colon [N] \to [N]$. (Recall from Eq. (3) that $\phi_{\mathrm{JW}}U_f = \Phi_{\mathrm{JW}}(U_f)\phi_{\mathrm{JW}}$.) By Theorem 2, there exists a circuit D

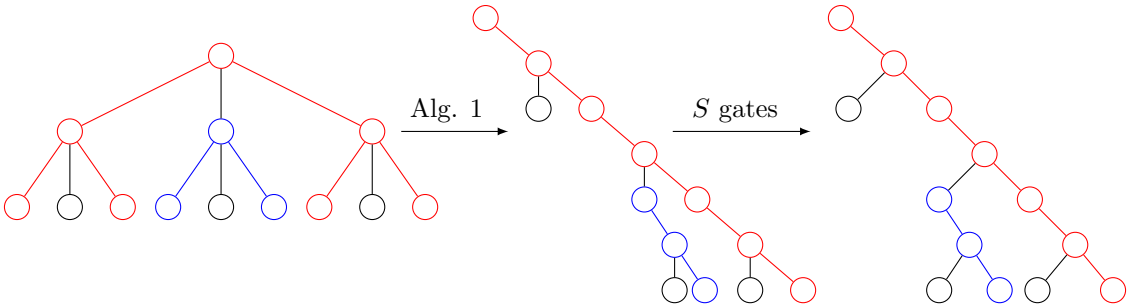


FIG. 15. In this example execution of Algorithm 3, we first apply Algorithm 2 on the red binary subtree as well as the blue binary subtree. Then we apply S gates on all middle nodes, depicted with a black edge, to transform them to left nodes. We would then apply Algorithm 2 again on the resulting binary tree. The qubit labelings and Majorana leaves are omitted.

that implements $\Phi_{\mathrm{JW}}(U_{\tau\sigma^{-1}})$ in depth $O(\log^2 N)$. Then $C_2^{-1}DC_1$ is a circuit of depth $O(\log^2 N)$ transforming from ϕ_1 to ϕ_2. □

Algorithm 3 Circuit transforming any ternary-tree shape into a right spine (Jordan-Wigner shape)

```
for all r ∈ {root node, every qubit node that is a middle child} do
    Apply Algorithm 2 to the binary subtree rooted at r
for all k ∈ {qubit nodes} do
    if k's middle child is not a leaf then
        Apply S_k
Apply Algorithm 2 on the resulting binary-shaped tree
```

Finally, we combine Theorems 2 and 5 to prove Theorem 1, which we now restate in technical language.

Theorem 1. *Given any product-preserving ternary tree mapping ϕ on N fermions and any permutation $\sigma\colon [N] \to [N]$, there exists a quantum circuit that implements $\Phi(U_\sigma)$ in depth $O(\log^2 N)$.*

Proof. By Theorem 5, there exists a circuit C of depth $O(\log^2 N)$ such that $\mathcal{C}\phi = \phi_{\mathrm{JW}}$. By Theorem 2, there exists a circuit D that implements $\Phi_{\mathrm{JW}}(U_\sigma)$ in depth $O(\log^2 N)$. Then $C^{-1}DC$ implements $\Phi(U_\sigma)$ in depth $O(\log^2 N)$, as desired. □

V. APPLICATIONS

In this section, we discuss applications of our methods to time evolution under widely studied fermionic models, see also Ref. [11] for applications of log-depth routing. We summarize our discussions in Table II, where we compare to the best previous methods that we are aware of.

Simulating fermionic Hamiltonians is one of the most promising applications of quantum computing, enabling the solution of classically challenging problems in quantum chemistry [30], materials science [31], and high-energy physics [32]. However, in order to simulate fermions with qubit quantum computers, the fermionic operators must be mapped to qubit operators. This leads to a multiplicative overhead in the circuit depth and gate count. In other words, this overhead is defined as the depth or gate count on a conventional qubit quantum computer (see e.g. Refs. [33–36] for recent experimental demonstrations) divided by the depth or gate count on a fermionic quantum computer that digitally manipulates fermionic particles directly (see e.g. Refs. [37–40]).

The simplest mapping of fermions to qubits is the Jordan-Wigner encoding [15]. One way to understand this mapping is that it maps products of fermionic operators to products of qubit operators. Naively, this encoding has worst-case depth overhead linear in N for N fermionic sites because some fermionic Hamiltonian terms are mapped to Pauli strings with support on $\Theta(N)$ sites, even if in the fermionic terms only have support on $\Theta(1)$ sites. This is because the Jordan-Wigner encoding labels the fermionic modes along a

fictitious one-dimensional chain. For example, the hopping term from one end of the chain to the other has weight 2 in the fermionic formulation and weight N after mapping to qubits. This is a highly geometrically non-local hopping term, but because of the effective one-dimensionality, even geometrically local hopping terms in low dimension can have high weight if they are not along the direction of the Jordan-Wigner chain. For example, the hopping term in two dimensions which is not along the chain has weight $O(\sqrt{N})$ [34]. While it is widely known that this leads to an at-worst linear depth overhead for fermion time evolution, the reason for this is often miscommunicated: it is not the high weight of the fermion terms which leads to the depth overhead, but the restriction in parallelization. The most straightforward approach for implementing time evolution under a Pauli-string Hamiltonian is done with a so-called Pauli gadget, i.e., the conjugation of a single-qubit rotation with an N-qubit CNOT ladder, which requires depth N. However, this CNOT ladder can actually be compressed to depth $O(\log N)$ without ancillas [12]. Therefore, a single fermionic Hamiltonian term can be implemented in depth $O(\log N)$ within the Jordan-Wigner encoding, and therefore the depth overhead resulting from the $O(N)$ operator weight is only $O(\log N)$. Instead, the reason why the Jordan-Wigner encoding has $O(N)$ depth overhead for the overall time-evolution task is the parallelization restriction: in the worst case, only one Hamiltonian term can be applied at a time, resulting in a depth scaling with the number of Hamiltonian terms, which is $\Omega(N)$ for a non-trivial Hamiltonian. By contrast, for fermionic quantum computers, there is no such restriction other than the connectivity graph of the Hamiltonian—in most physical Hamiltonians, the connectivity is such that $\Theta(N)$ terms can be implemented in parallel (this is the case in the 2D Fermi-Hubbard model [41]), such that for example $\Theta(N)$ Hamiltonian terms can be implemented with depth $\Theta(1)$.

The Bravyi-Kitaev encoding [14] might at first seem to address this issue [16], as it reduces the operator weight after the fermion-to-qubit mapping to only $O(\log N)$ sites. However, unfortunately, these savings for a single fermionic Hamiltonian term do not carry over to the task of implementing all terms: the Pauli strings to which products of fermion creation and annihilation operators are mapped all have one qubit in common, so they cannot be implemented in parallel [42]. Therefore, in practice, even though the weight reduction can lead to a gate count improvement, for most problems the Bravyi-Kitaev encoding does not show an improvement over the $O(N)$ depth overhead of the Jordan-Wigner encoding. Schemes have been proposed to avoid this restriction by using ancillas [14, 39, 42–44], which in particular can achieve $O(1)$ overhead for geometrically local models [20, 45]. However, all of these approaches require $\Omega(N)$ ancillas, which might be a prohibitive overhead, especially for fault-tolerant architectures, where qubit count is often more expensive than depth due to the exponential error suppression [46].

Instead, we here rely on the polylog routing schemes from Ref. [11] and this work to reduce the depth of fermion time evolution.

Reducing the fermion-to-qubit overhead in fermion time evolution to permutations

Our aim is to implement a product-formula approximation of the time evolution under a fermionic Hamiltonian H with minimal circuit depth. For concreteness and simplicity, consider the Hamiltonian $H = \sum_{i,j=1}^{N} J_{ij}(a_i^\dagger a_j + \text{h.c.})$. We consider a single-step first-order scheme that approximates the time-evolution operator $U = \exp(-iHt)$ as

$$U = \prod_{i,j=1}^{N} \exp(-itJ_{ij}(a_i^\dagger a_j + \text{h.c.})) + O(t^2). \tag{13}$$

As we will discuss near the end of this section, both of these restrictions are not necessary: this scheme generalizes to any Trotter scheme and any Hamiltonian (without any restrictions on k-locality).

On a fermionic quantum computer, which in particular can directly implement $\exp(-itJ_{ij}(a_i^\dagger a_j + \text{h.c.}))$ as a single gate, the minimal depth to implement this scheme depends only on how many of the J_{ij} are nonzero and to what extent the structure of the J_{ij} may be parallelized. For instance, for a full-rank tensor, i.e., with $\Theta(N^2)$ non-zero terms, the terms can be grouped such that $O(N)$ of them can be implemented in parallel. This means that the minimal depth is $\Omega(N)$. For a non-full-rank tensor, the situation is more complicated: if there are only $\Theta(N)$ non-zero terms, this may still require depth $\Theta(N)$ if all of the terms have one fermion in common. In order to discuss the overhead introduced by the fermion-to-qubit encoding independently of the parallelization restrictions introduced by the Hamiltonian parameters, we define the fermion-to-qubit overhead as the multiplicative overhead in gate depth between an implementation on a fermionic quantum

computer and a qubit quantum computer, or in other words, the gate depth on a qubit quantum computer divided by the gate depth on a fermion quantum computer, as we have already defined above. To make this comparison, we assume all-to-all connectivity in both qubit and fermion quantum computers.

One of the most efficient ways to perform time evolution in the Jordan-Wigner encoding are the fSWAP networks introduced in Ref. [10]. They have been shown to achieve $O(N)$ depth (and more generally, $O(N^{k-1})$ for k-local Hamiltonians Ref. [47]) for the simulation task in Eq. (13) by applying transpositions of neighboring fermions along the Jordan-Wigner chain between fermion gate layers, enabling every possible ordering of the fermions to be traversed with only constant depth overhead. Therefore, fSWAP networks are optimal for all-to-all connected Hamiltonians. However, there is a large class of Hamiltonians with $\Theta(N)$ non-zero J_{ij} that can be implemented in depth $O(1)$ on fermionic quantum computers, but whose implementation with fSWAP networks on qubit quantum computers still uses depth scaling as some power of the system size. One example is the nearest-neighbor Fermi-Hubbard model in two spatial dimensions, for which current fSWAP implementations use $\Theta(\sqrt{N})$ depth [34].

In this section, we discuss that, using the schemes introduced in Ref. [11] and this work, fSWAP networks can in fact achieve a worst-case upper bound of $O(\log^2 N)$ depth overhead compared to fermionic quantum computers for *general* fermionic Hamiltonians without introducing ancilla qubits. We do so by simply routing the fermion along the Jordan-Wigner chain in-between layers of time evolutions that only act on neighboring fermionic sites, in complete analogy to how routing helps in reducing the depth of all-to-all-connected circuits in 1D nearest-neighbour-connected hardware [8].

Specifically, for our example, to order t^2,

$$\begin{aligned} U &\approx \prod_{l=1}^{N_{\text{layers}}} U_{\sigma_l}^{-1} U_{\sigma_l} \prod_{J_{ij}\in \text{layer } l} \exp(-itJ_{ij}(a_i^\dagger a_j + \text{h.c.})) U_{\sigma_l}^{-1} U_{\sigma_l} \\ &= \prod_{l=1}^{N_{\text{layers}}} U_{\sigma_l}^{-1} \prod_{J_{ij}\in \text{layer } l} \exp(-itJ_{ij}(a_{\sigma_l(i)}^\dagger a_{\sigma_l(j)} + \text{h.c.})) U_{\sigma_l}. \end{aligned} \tag{14}$$

(We define the product of operators using the following order: $\prod_{i=1}^{N} O_i = O_N \cdots O_2 O_1$.) This scheme, see also Ref. [11], is a generalization of the time evolution scheme in the original work introducing fSWAP networks [10] and is also closely related to the double factorization scheme [48], where fermion-overhead-free density-density interactions are alternated with fermionic basis transformations. One way to understand the above scheme is that part of the basis transformation in double factorization is absorbed in the Hamiltonian (such that now it is not purely density-density any more), and the rest of the basis transformation is restricted to permutations of fermions.

The terms are grouped into N_{layers} layers, so that terms within a given layer can be applied in parallel on a fermionic quantum computer. The goal of the permutations is to rearrange the order of the fermion sites in a given layer so that the Hamiltonian terms only act on sites that are adjacent in the Jordan-Wigner encoding, such that k-local fermion terms map to k-local qubit terms. In the first line of Eq. (14), we insert resolutions of identity in terms of permutation unitaries U_{σ_l}, which apply permutation σ_l corresponding to layer l: $U_{\sigma_l} a_i U_{\sigma_l}^{-1} = a_{\sigma_l(i)}$. To enable parallelization of all terms within a given layer in the Jordan-Wigner encoding, the permutation unitaries U_{σ_l} reorder the fermions along the Jordan-Wigner chain such that, for all J_{ij} within a given layer, the fermions $\sigma_l(i)$ and $\sigma_l(j)$ are neighboring along the Jordan-Wigner chain. This means that $\exp(-itJ_{ij}(a_{\sigma_l(i)}^\dagger a_{\sigma_l(j)} + \text{h.c.})) \to \exp(-itJ_{ij}(\Sigma_{\sigma_l(i)}^+ \Sigma_{\sigma_l(j)}^- + \text{h.c.}))$, where $\Sigma^\pm = (X \pm iY)/2$ are qubit Pauli operators, can be implemented in depth $O(1)$ and no parallelization restriction arises from the fermion-to-qubit mapping. Furthermore, for every $l < N_{\text{layers}}$, $U_{\sigma_{l+1}} U_{\sigma_l}^{-1}$ can be combined into a single permutation unitary.

Our discussion above straightforwardly generalizes to any fermionic Hamiltonian (without any restrictions on spacial locality or k-locality but with terms that contain an even number of fermionic creation and annihilation operators) as we can in any case, in particular also for an arbitrary order product formula, write time evolution as alternations of fermion permutations and time evolutions in such a way that the time evolution only acts on Jordan-Wigner adjacent modes. For example, for a 4-local term $\propto a_i^\dagger a_j^\dagger a_k a_m$ as appearing in quantum chemistry, the permutations simply exchange four fermion operators such that after the permutations, they are adjacent along the Jordan-Wigner chain and we can map $a_{\sigma_l(i)}^\dagger a_{\sigma_l(j)}^\dagger a_{\sigma_l(k)} a_{\sigma_l(m)} \to \Sigma_{\sigma_l(i)}^+ \Sigma_{\sigma_l(j)}^+ \Sigma_{\sigma_l(k)}^- \Sigma_{\sigma_l(m)}^-$. The resulting at most $N/4$ terms can then be applied without any parallelization restrictions.

As an example, we consider simulation of fermionic Hamiltonians of the form

$$H = \sum_{i<j} J_{ij}(a_i^\dagger a_j + \text{h.c.}) + \sum_{i<j} U_{ij} n_i n_j, \tag{15}$$

where $n_i = a_i^\dagger a_i$, and J_{ij}, U_{ij} are the hopping and interaction coefficient matrices, respectively. This problem is relevant to simulations of materials [31].

Fermionic fast Fourier transform. The fermionic fast Fourier transform [49] diagonalizes the hopping Hamiltonian provided J_{ij} is translationally invariant on a d-dimensional hypercubic lattice with periodic boundary conditions (we focus on the 1-dimensional case here since the d-dimensional FFFT is simply the one-dimensional FFFT applied in all dimensions [31], just as for the usual FFT). Concretely,

$$\sum_{i<j} J_{ij}(a_i^\dagger a_j + \text{h.c.}) = \text{FFFT} \left(\sum_i \theta_i n_i \right) \text{FFFT}^\dagger, \tag{16}$$

where the θ_i can be computed classically. This means that the depth of evolving under $\sum_{i<j} J_{ij}(a_i^\dagger a_j + \text{h.c.})$ is twice the depth of the FFFT plus the depth of evolving under $\sum_i \theta_i n_i$, which is $O(1)$ in the Jordan-Wigner encoding.

In the following, we discuss that the FFFT can be applied in polylog depth, implying that a single Trotter step under the hopping Hamiltonian for translationally-invariant systems can be done in polylog depth, as shown in Ref. [11], see also Ref. [39, 49] for the corresponding statement for fermionic quantum computers.

Corollary 2 (Ancilla-free polylog-depth FFFT)**.** *The fermionic fast Fourier transform [45] on N fermionic modes can be implemented in depth $O(\log^2 N)$ in the Jordan-Wigner encoding.*

This means the depth overhead is only $O(\log N)$ as compared to a fermionic quantum computer [39, 45].

Proof. Inspecting the circuit for implementing the FFFT (see Fig. 1 in Ref. [45]) reveals that it consists of $O(\log N)$ fSWAP layers and $O(\log N)$ layers of parallel interactions. Each interaction layer can be performed in depth $O(1)$ in the Jordan-Wigner encoding because each interaction involves only two modes adjacent in the Jordan-Wigner ordering. In addition, inspecting each fSWAP layer reveals that each is a staircase permutation, with the exception of the last layer. Because staircase permutations can be performed in depth $O(\log N)$ and general permutations in depth $O(\log^2 N)$ (per Theorem 2), the FFFT can be performed in depth $O(\log^2 N)$. □

Finally, while the focus of this work is on ancilla-free circuits, we can remove one factor of $\log N$ if we allow for ancillas and mid-circuit measurements:

Corollary 3 (Log-depth FFFT with ancillas [11])**.** *Given any permutation $\sigma\colon [N] \to [N]$, there exists a circuit of depth $O(\log N)$ with $O(N)$ ancillas and measurement and feedforward implementing $\Phi_{\text{JW}}(U_\sigma)$. In addition, there is a circuit of depth $O(\log N)$ implementing the FFFT using $O(N)$ ancillas and measurement and feedforward.*

Proof. The final circuit for performing a fermionic staircase permutation in Figs. 9 and 10 is composed of three contiguous-range CZ fanouts, each consisting of two queries to the parity transform and several CZ-fanout operators on independent qubits. CZ-fanout and P (which is equivalent to a CNOT ladder) can both be implemented in constant depth [50, 51] using a constant number of rounds of measurements and $O(N)$ ancillas, reducing each of the logarithmically many layers of staircase permutations to constant depth. In addition, as mentioned in Corollary 2, each permutation step in the FFFT implementation is a staircase permutation, with the exception of the last, showing that the depth of the entire FFFT circuit is reduced to $O(\log N)$ with $O(N)$ ancillas and measurements. □

Next, we use this efficient FFFT to reduce the circuit depth in important models.

(Sub-)routine	Compact [17, 20]		fSWAP [10]		pfSWAP [54]		Maskara et al. [11]		This work	
	Depth	Anc.	Depth	Anc.	Depth	Anc.	Depth	Anc.	Depth	Anc.
Fermion permut.	N	N	N	0	1	N^2	$\log N$	N	$\log^2 N$	0
FFFT	$N \log N$	N	$N \log N$	0	$\log N$	N^2	$\log N$	N	$\log^2 N$	0
Materials (1 step)	N	N	N	0	$\log N$	N^2	$\log N$	$N \log N$	$\log^2 N$	$N \log N$
Materials (full)	N^2	N	N^2	0	$N \log N$	N^2	$N \log N$	$N \log N$	$N \log^2 N$	$N \log N$
2D Fermi-Hubbard	1	N	$\sqrt{N}$	0	$\log N$	N^2	$\log N$	N	$\log^2 N$	0

TABLE II. Asymptotic circuit depth and ancilla count for key fermionic subroutines and applications. All quantities are $O(\cdot)$ upper bounds. Compact refers to any compact encoding, e.g., Verstraete-Cirac [17] or Derby-Klassen [20]; fSWAP is the transposition-based fSWAP network from Ref. [10]; pfSWAP refers to parallel fSWAP from Ref. [39] using $O(N^2)$ ancillas to brute-force parallelize the CZ circuits in the fSWAP networks; "2D Fermi-Hubbard" applies, more generally, to any model with $O(1)$-range hopping and interactions and assumes periodic boundary conditions for pfSWAP and our work (as we assume the usage of the FFFT for the kinetic term), and the gate counts are for a single Trotter step. The number of Trotter steps $O(N)$ used in "Materials (full)" assumes arbitrarily large Trotter order, fixed evolution time, and fixed simulation error. Maskara et al. [11] makes use of ancillas in the permutation step, whereas our work only uses ancillas for parallelizing the interaction part of the Hamiltonian [52].

Fermi-Hubbard-like models. We first discuss the two-dimensional Fermi-Hubbard model as an example for which our encoding yields a depth advantage when disallowing ancillas. In this scenario, the J_{ij} coefficients are only non-zero for finite-range hopping on a two-dimensional lattice. Specifically, J_{ij} is non-zero only if site j is within a finite number of cartesian units from site i on the 2D lattice. We also make the same assumption about the U_{ij}. In this case, one way to implement Trotter time evolution is to first apply the hopping terms, and then the interactions (or vice versa). Each of those two components is divided into $O(1)$ groups of hopping/interaction terms that only act on two sites such that all $O(N)$ terms in that group can be applied in parallel. For example, for nearest-neighbor hopping in 2D, there are four groups: two each for the hopping along the two directions of the lattice. This means that, on a fermionic quantum computer, a single Trotter step can be applied in $O(1)$ depth. Compact encodings [17, 20] achieve the same scaling by introducing $O(N)$ ancillas and preparing a toric-code state in the beginning of the calculation, which can be done in depth $O(1)$ using midcircuit measurement and feedforward. Therefore, compact encodings achieve $O(1)$ depth overhead.

Our scheme does not achieve $O(1)$ depth, even with $O(N)$ ancillas, and is therefore asymptotically inferior when $O(N)$ ancillas are allowed. However, the lowest-depth ancilla-free scheme known so far uses fSWAP networks based on local transpositions, achieving $O(\sqrt{N})$ depth [10, 34]. We can improve on this in cases in which the Hamiltonian is translationally invariant: in that case, we can employ the FFFT to implement time evolution under the hopping part, leading to $O(\log^2 N)$ depth. Because the interaction term can be applied in depth $O(1)$, this implies overall depth $O(\log^2 N)$ for a single Trotter step.

Materials simulation. The materials Hamiltonian—or equivalently, quantum chemistry in the plane-wave dual basis [31]—is essentially a more complicated version of the Fermi-Hubbard model: the coefficients J_{ij} and U_{ij} are full rank, but both are translationally invariant [31]. Therefore, the hopping part can be compressed again to $O(\log^2 N)$ depth without ancillas and $O(\log N)$ depth with ancillas by using the FFFT and polylog-depth permutations. By contrast, because of the unboundedness of the hoppings, no scheme is known how to achieve this depth in the compact encoding. Instead, the depth in the compact encoding defaults to the same asymptotic depth as using fSWAP networks with transpositions [10] as the same geometric restrictions apply. A scheme to reduce the depth to $O(\log N)$ was proposed in Ref. [39], but required $O(N^2)$ ancillas. For the interaction term, no scheme is known to apply it in polylog depth without ancillas; however, an $O(\log N)$-depth scheme is known using $O(N \log N) = \tilde{O}(N)$ ancillas [52]. Therefore, the schemes proposed in Ref. [11] and in this work achieve polylog depth for a single Trotter step, as already noted in [11]. Finally, for obtaining the end-to-end depth for simulating time evolution until time T, we need to multiply this by the number of Trotter steps. The best known upper bound is $O(NT)$ from Ref. [53], assumes arbitrarily large Trotter order and fixed total simulation error. Therefore, our schemes achieve $\tilde{O}(NT)$ depth complexity with $\tilde{O}(N)$ qubits. This quasilinear-in-number-of-plane-waves scaling of qubits and depth has to our knowledge not been achieved for qubit quantum computers yet and this scaling is on par (asymptotically) with fermionic quantum computers [39].

VI. DISCUSSION AND OUTLOOK

In this work, we build on the recent exponential improvement in fermionic routing introduced by Maskara et al. [11], who showed that the worst-case routing overhead in fermion-to-qubit mappings can be reduced from $O(N)$ to $O(\log N)$ by decomposing general fermionic permutations into $O(\log N)$ constant-depth interleaves and using $\Theta(N)$ ancilla qubits and mid-circuit measurements to compress the CZ circuits part of fermi-SWAP networks [10]. We rederive and generalize their results using an alternative construction. Moreover, we show that similar exponential improvements persist even in the ancilla-free setting: any fermionic permutation in the Jordan-Wigner encoding can be implemented in depth $O(\log^2 N)$ without ancillas. We further extend our analysis to arbitrary product-preserving ternary-tree encodings by constructing efficient mappings between those encodings and the Jordan-Wigner encoding in $O(\log^2 N)$ depth. This establishes that fermionic routing can be efficiently performed in a broad class of fermionic encodings. As a result, key primitives such as the fermionic fast Fourier transform (FFFT) and single-Trotter-step material simulations can be executed in polylogarithmic depth using only $\tilde{O}(N)$ qubits for N fermionic modes.

Our results assume hardware with all-to-all connectivity, but they also imply upper bounds for limited-connectivity devices. In particular, the worst-case overhead for fermionic simulation in constrained architectures is $O(\log^2 N)$ times the worst-case depth required to route qubits on the underlying hardware graph, if that hardware graph has $\Omega(N)$ disjoint edges [55]. For instance, qubit systems with grid connectivity can implement the FFFT in depth $O(\sqrt{N}\log^2 N)$; current neutral-atom architectures achieve $O(\sqrt{N}\log^3 N)$ depth, while upgraded neutral-atom hardware could reach $O(\log^3 N)$ depth [8]. This represents an exponential improvement over the prior best-known FFFT algorithm, which achieved $O(N \log N)$ depth [31] on a linear geometry, with no mechanism to exploit higher connectivity. The above-cited depths in the presence of hardware constraints is likely too pessimistic as it neglects additional structure in the circuits. In fact, the FFFT is an example where that structure can be exploited, and Maskara et al. [11] showed several others.

Finally, these results motivate further study of fermionic routing and simulation on hardware with restricted connectivity, particularly the trade-offs among ancilla count, mid-circuit measurement and feedforward, and depth overhead. Beyond asymptotic scaling, a detailed analysis of constant factors in depth and gate count will be essential for identifying the most practical regimes of advantage. It also remains an open question whether analogous circuit-structural insights could lead to improved boson-to-qubit encodings [56–58].

ACKNOWLEDGMENTS

We thank Maskara et al. for sharing their results ahead of posting their manuscript [11] on the arXiv.

N.C., J.Y., D.D., A.F., A.S., and A.V.G. acknowledge support by the U.S. Department of Energy, Office of Science, National Quantum Information Science Research Centers, Quantum Systems Accelerator (QSA). A.M.C., N.C., J.Y., D.D., M.J.G., A.F., A.S., and A.V.G. acknowledge support by NSF QLCI (award No. OMA-2120757), DoE ASCR Quantum Testbed Pathfinder program (awards No. DE-SC0019040 and No. DE-SC0024220), and the U.S. Department of Energy, Office of Science, Accelerated Research in Quantum Computing, Fundamental Algorithmic Research toward Quantum Utility (FAR-Qu). N.C., J.Y., D.D., A.F., A.S., and A.V.G. acknowledge support by ONR MURI, NSF STAQ program, AFOSR MURI, DARPA SAVaNT ADVENT, ARL (W911NF-24-2-0107), and NQVL:QSTD:Pilot:FTL.

[1] S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington, and R. Duncan, t|ket⟩: a retargetable compiler for nisq devices, Quantum Sci. Technol. **6**, 014003 (2020).

[2] A. M. Childs, E. Schoute, and C. M. Unsal, Circuit transformations for quantum architectures, in *14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2019)*, Leibniz International Proceedings in Informatics (LIPIcs), Vol. 135 (Dagstuhl, Germany, 2019) pp. 3:1–3:24.

[3] A. Cowtan, S. Dilkes, R. Duncan, A. Krajenbrink, W. Simmons, and S. Sivarajah, On the qubit routing problem, in *14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2019)*, Leibniz International Proceedings in Informatics (LIPIcs), Vol. 135 (Dagstuhl, Germany, 2019) pp. 5:1–5:32.

[4] A. Plathanam Babu, O. Kerppo, A. Muñoz-Moller, M. Haghparast, and M. Silveri, Gate teleportation-assisted routing for quantum algorithms, Quantum Sci. Technol. **10**, 035004 (2025).

[5] G. Padda, E. Tham, A. Brodutch, and D. Touchette, Improving qubit routing by using entanglement mediated remote gates, in *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, Vol. 01 (2024) p. 1770–1776.

[6] D. Devulapalli, E. Schoute, A. Bapat, A. M. Childs, and A. V. Gorshkov, Quantum routing with teleportation, Phys. Rev. Res. **6**, 033313 (2024).

[7] S. Moses, C. Baldwin, M. Allman, R. Ancona, L. Ascarrunz, C. Barnes, J. Bartolotta, B. Bjork, P. Blanchard, M. Bohn, J. Bohnet, N. Brown, N. Burdick, W. Burton, S. Campbell, J. Campora, C. Carron, J. Chambers, J. Chan, Y. Chen, A. Chernoguzov, E. Chertkov, J. Colina, J. Curtis, R. Daniel, M. DeCross, D. Deen, C. Delaney, J. Dreiling, C. Ertsgaard, J. Esposito, B. Estey, M. Fabrikant, C. Figgatt, C. Foltz, M. Foss-Feig, D. Francois, J. Gaebler, T. Gatterman, C. Gilbreth, J. Giles, E. Glynn, A. Hall, A. Hankin, A. Hansen, D. Hayes, B. Higashi, I. Hoffman, B. Horning, J. Hout, R. Jacobs, J. Johansen, L. Jones, J. Karcz, T. Klein, P. Lauria, P. Lee, D. Liefer, S. Lu, D. Lucchetti, C. Lytle, A. Malm, M. Matheny, B. Mathewson, K. Mayer, D. Miller, M. Mills, B. Neyenhuis, L. Nugent, S. Olson, J. Parks, G. Price, Z. Price, M. Pugh, A. Ransford, A. Reed, C. Roman, M. Rowe, C. Ryan-Anderson, S. Sanders, J. Sedlacek, P. Shevchuk, P. Siegfried, T. Skripka, B. Spaun, R. Sprenkle, R. Stutz, M. Swallows, R. Tobey, A. Tran, T. Tran, E. Vogt, C. Volin, J. Walker, A. Zolot, and J. Pino, A race-track trapped-ion quantum processor, Phys. Rev. X **13**, 041052 (2023).

[8] N. Constantinides, A. Fahimniya, D. Devulapalli, D. Bluvstein, M. J. Gullans, J. V. Porto, A. M. Childs, and A. V. Gorshkov, Optimal routing protocols for reconfigurable atom arrays (2024), arXiv:2411.05061.

[9] Q. Xu, J. P. Bonilla Ataides, C. A. Pattison, N. Raveendran, D. Bluvstein, J. Wurtz, B. Vasić, M. D. Lukin, L. Jiang, and H. Zhou, Constant-overhead fault-tolerant quantum computation with reconfigurable atom arrays, Nat. Phys. **20**, 1084 (2024).

[10] I. D. Kivlichan, J. McClean, N. Wiebe, C. Gidney, A. Aspuru-Guzik, G. K.-L. Chan, and R. Babbush, Quantum simulation of electronic structure with linear depth and connectivity, Phys. Rev. Lett. **120**, 110501 (2018).

[11] N. Maskara, M. Kalinowski, D. Gonzalez-Cuadra, and M. D. Lukin, Fast simulation of fermions with reconfigurable qubits (2025), arXiv:2509.08898v1.

[12] M. Remaud and V. Vandaele, Ancilla-free quantum adder with sublinear depth, in *Reversible Computation* (Springer Nature Switzerland, 2025) p. 137–154.

[13] Nishad Maskara, private communication.

[14] S. B. Bravyi and A. Y. Kitaev, Fermionic quantum computation, Ann. Phys. **298**, 210–226 (2002).

[15] P. Jordan and E. P. Wigner, Über das Paulische Äquivalenzverbot, Z. Phys. **47**, 631 (1928).

[16] J. T. Seeley, M. J. Richard, and P. J. Love, The Bravyi-Kitaev transformation for quantum computation of electronic structure, J. Chem. Phys. **137**, 224109 (2012).

[17] F. Verstraete and J. I. Cirac, Mapping local Hamiltonians of fermions to local Hamiltonians of spins, J. Stat. Mech. **2005**, P09012 (2005).

[18] K. Setia, S. Bravyi, A. Mezzacapo, and J. D. Whitfield, Superfast encodings for fermionic quantum simulation, Phys. Rev. Res. **1**, 033033 (2019).

[19] Z. Jiang, A. Kalev, W. Mruczkiewicz, and H. Neven, Optimal fermion-to-qubit mapping via ternary trees with applications to reduced quantum states learning, Quantum **4**, 276 (2020).

[20] C. Derby, J. Klassen, J. Bausch, and T. Cubitt, Compact fermion to qubit mappings, Phys. Rev. B **104**, 035118 (2021).

[21] C. Moore, Quantum circuits: Fanout, parity, and counting (1999), arXiv:quant-ph/9903046.

[22] M. Fang, S. Fenner, F. Green, S. Homer, and Y. Zhang, Quantum lower bounds for fanout, Quantum Info. Comput. **6**, 46–57 (2006).

[23] A. Y. Vlasov, Clifford algebras, spin groups and qubit trees, Quanta **11**, 97–114 (2022).

[24] A. Miller, Z. Zimborás, S. Knecht, S. Maniscalco, and G. García-Pérez, Bonsai algorithm: Grow your own fermion-to-qubit mappings, PRX Quantum **4**, 030314 (2023).

[25] M. Chiew, B. Harrison, and S. Strelchuk, Ternary tree transformations are equivalent to linear encodings of the Fock basis (2024), arXiv:2412.07578.

[26] J. Yu, Y. Liu, S. Sugiura, T. Van Voorhis, and S. Zeytinoğlu, Clifford circuit-based heuristic optimization of fermion-to-qubit mappings, J. Chem. Theory Comput. **21**, 9430 (2025).

[27] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. (MIT Press, 2009).

[28] A. Y. Vlasov, Mutual transformations of arbitrary ternary qubit trees by Clifford gates (2024), arXiv:2404.16693.

[29] N. Alon, F. R. K. Chung, and R. L. Graham, Routing permutations on graphs via matchings, SIAM J. Discrete Math. **7**, 513 (1994).

[30] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan, Quantum computational chemistry, Rev. Mod. Phys. **92**, 015003 (2020).

[31] R. Babbush, N. Wiebe, J. McClean, J. McClain, H. Neven, and G. K.-L. Chan, Low-Depth Quantum Simulation of Materials, Phys. Rev. X **8**, 011044 (2018).

[32] C. W. Bauer, Z. Davoudi, A. B. Balantekin, T. Bhattacharya, M. Carena, W. A. de Jong, P. Draper, A. El-Khadra, N. Gemelke, M. Hanada, D. Kharzeev, H. Lamm, Y.-Y. Li, J. Liu, M. Lukin, Y. Meurice, C. Monroe, B. Nachman, G. Pagano, J. Preskill, E. Rinaldi, A. Roggero, D. I. Santiago, M. J. Savage, I. Siddiqi, G. Siopsis,

D. Van Zanten, N. Wiebe, Y. Yamauchi, K. Yeter-Aydeniz, and S. Zorzetti, Quantum Simulation for High-Energy Physics, PRX Quantum **4**, 027001 (2023).
[33] S. Stanisic, J. L. Bosse, F. M. Gambetta, R. A. Santos, W. Mruczkiewicz, T. E. O'Brien, E. Ostby, and A. Montanaro, Observing ground-state properties of the Fermi-Hubbard model using a scalable algorithm on a quantum computer, Nat Commun **13**, 5743 (2022).
[34] K. Hémery, K. Ghanem, E. Crane, S. L. Campbell, J. M. Dreiling, C. Figgatt, C. Foltz, J. P. Gaebler, J. Johansen, M. Mills, S. A. Moses, J. M. Pino, A. Ransford, M. Rowe, P. Siegfried, R. P. Stutz, H. Dreyer, A. Schuckert, and R. Nigmatullin, Measuring the Loschmidt Amplitude for Finite-Energy Properties of the Fermi-Hubbard Model on an Ion-Trap Quantum Computer, PRX Quantum **5**, 030323 (2024).
[35] R. Nigmatullin, K. Hémery, K. Ghanem, S. Moses, D. Gresh, P. Siegfried, M. Mills, T. Gatterman, N. Hewitt, E. Granet, and H. Dreyer, Experimental demonstration of breakeven for a compact fermionic encoding, Nat. Phys. , 1 (2025).
[36] S. J. Evered, M. Kalinowski, A. A. Geim, T. Manovitz, D. Bluvstein, S. H. Li, N. Maskara, H. Zhou, S. Ebadi, M. Xu, J. Campo, M. Cain, S. Ostermann, S. F. Yelin, S. Sachdev, M. Greiner, V. Vuletić, and M. D. Lukin, Probing the Kitaev honeycomb model on a neutral-atom quantum computer, Nature **645**, 341 (2025).
[37] Z. Z. Yan, B. M. Spar, M. L. Prichard, S. Chi, H.-T. Wei, E. Ibarra-García-Padilla, K. R. A. Hazzard, and W. S. Bakr, Two-Dimensional Programmable Tweezer Arrays of Fermions, Phys. Rev. Lett. **129**, 123201 (2022).
[38] D. González-Cuadra, D. Bluvstein, M. Kalinowski, R. Kaubruegger, N. Maskara, P. Naldesi, T. V. Zache, A. M. Kaufman, M. D. Lukin, H. Pichler, B. Vermersch, J. Ye, and P. Zoller, Fermionic quantum processing with programmable neutral atom arrays, Proc. Natl. Acad. Sci. **120**, e2304294120 (2023).
[39] A. Schuckert, E. Crane, A. V. Gorshkov, M. Hafezi, and M. J. Gullans, Fermion-qubit fault-tolerant quantum computing (2024), arXiv:2411.08955.
[40] R. Ott, D. González-Cuadra, T. V. Zache, P. Zoller, A. M. Kaufman, and H. Pichler, Error-corrected fermionic quantum processors with neutral atoms (2024), arXiv:2412.16081.
[41] E. T. Campbell, Early fault-tolerant simulations of the Hubbard model, Quantum Sci. Technol. **7**, 015007 (2021).
[42] J. Bringewatt and Z. Davoudi, Parallelization techniques for quantum simulation of fermionic systems, Quantum **7**, 975 (2023).
[43] R. W. Chien and J. D. Whitfield, Custom fermionic codes for quantum simulation (2020), arXiv:2009.11860.
[44] M. Luo and J. I. Cirac, Efficient simulation of quantum chemistry problems in an enlarged basis set (2025), arXiv:2407.04432.
[45] F. Verstraete, J. I. Cirac, and J. I. Latorre, Quantum circuits for strongly correlated quantum systems, Phys. Rev. A **79**, 032316 (2009).
[46] C. Gidney, How to factor 2048 bit RSA integers with less than a million noisy qubits (2025), arXiv:2505.15917.
[47] B. O'Gorman, W. J. Huggins, E. G. Rieffel, and K. B. Whaley, Generalized swap networks for near-term quantum computing (2019), arXiv:1905.05118.
[48] M. Motta, E. Ye, J. R. McClean, Z. Li, A. J. Minnich, R. Babbush, and G. K.-L. Chan, Low rank representations for quantum simulation of electronic structure, npj Quantum Inf **7**, 1 (2021).
[49] F. Verstraete, J. I. Cirac, and J. I. Latorre, Quantum circuits for strongly correlated quantum systems, Phys. Rev. A **79**, 032316 (2009).
[50] E. Bäumer and S. Woerner, Measurement-based long-range entangling gates in constant depth, Phys. Rev. Res. **7**, 023120 (2025).
[51] N. Cody Jones, J. D. Whitfield, P. L. McMahon, M.-H. Yung, R. V. Meter, A. Aspuru-Guzik, and Y. Yamamoto, Faster quantum chemistry simulation on fault-tolerant quantum computers, New J. Phys. **14**, 115023 (2012).
[52] G. H. Low and N. Wiebe, Hamiltonian Simulation in the Interaction Picture (2019), arXiv:1805.00675.
[53] A. M. Childs, Y. Su, M. C. Tran, N. Wiebe, and S. Zhu, Theory of Trotter Error with Commutator Scaling, Phys. Rev. X **11**, 011020 (2021).
[54] A. Schuckert, E. Crane, A. V. Gorshkov, M. Hafezi, and M. J. Gullans, Fault-tolerant fermionic quantum computing (2025), arXiv:2411.08955.
[55] P. Yuan and S. Zhang, Full Characterization of the Depth Overhead for Quantum Circuit Compilation with Arbitrary Qubit Connectivity Constraint, Quantum **9**, 1757 (2025).
[56] N. P. D. Sawaya, T. Menke, T. H. Kyaw, S. Johri, A. Aspuru-Guzik, and G. G. Guerreschi, Resource-efficient digital quantum simulation of d-level systems for photonic, vibrational, and spin-s Hamiltonians, npj Quantum Inf **6**, 49 (2020).
[57] E. Crane, K. C. Smith, T. Tomesh, A. Eickbusch, J. M. Martyn, S. Kühn, L. Funcke, M. A. DeMarco, I. L. Chuang, N. Wiebe, A. Schuckert, and S. M. Girvin, Hybrid Oscillator-Qubit Quantum Processors: Simulating Fermions, Bosons, and Gauge Fields (2024), arXiv:2409.03747.
[58] Y. Liu, S. Singh, K. C. Smith, E. Crane, J. M. Martyn, A. Eickbusch, A. Schuckert, R. D. Li, J. Sinanan-Singh, M. B. Soley, T. Tsunoda, I. L. Chuang, N. Wiebe, and S. M. Girvin, Hybrid Oscillator-Qubit Quantum Processors: Instruction Set Architectures, Abstract Machine Models, and Applications (2025), accepted in PRX Quantum Tutorials, arXiv:2407.10381.