Chapter 3

Embedding simulations with quantum codes

3.1 Background

While small quantum simulations have been performed on few-qubit devices across all platforms [24, 41, 44–48], and efforts are undertaken to scale devices up, the quantum simulation of larger fermionic systems is still a challenge. Critical factors that determine the feasibility of an algorithm would be its qubit requirements, its gate cost (in terms of magic states when error-corrected, and in terms of two-qubit gates when noisy) [28, 49] and circuit depth (a measure of the algorithm run time, where each time step is the duration of one quantum gate). Quantum algorithms are generally to be kept shallow to ensure that they can be run before the qubit system has decohered. It is thus in our interest to decompose the algorithms into many parts that can be run in parallel, i.e. at the same time. Obviously, one can hope for parallelization if the algorithm is comprised of gate sequences that act on subsets of as few qubits as possible and these subsets do not overlap much. Another factor is that actual quantum devices can have geometric limitations which negatively influence the circuit depth. In a practical setting not every qubit can reach every other qubit, i.e. they cannot be entangled with a single two-qubit gate. To entangle distant qubits, it takes additional efforts in gates and time. Thus another criterion for the reduction of the circuit depth is that gate sequences only act on qubits adjacent on a certain connectivity graph. Although this graph depends on the actual quantum

device, we can make an educated guess: devices on which surface code can be run, require a square lattice connectivity graph.
Unfortunately, it is nontrivial to embed fermionic problems in those lattices, which opposes shallow-depth quantum simulation. Let us illustrate the exact issue. In order to bring the problem into a form the quantum computer can process, the fermionic modes need to be embedded into a (two-dimensional) lattice structure related to the qubit connectivity graph. After that, a fermion-to-qubit mapping translates the interactions of those system to a qubit Hamiltonian fit to be simulated. It is this last step in which the problem lies, as simulating the interaction between as little as two fermionic modes usually requires gates acting on large subsets of qubits. This is a consequence of the fermionic wave functions being antisymmetric under particle permutations, which causes the interaction of two fermionic modes to also be sensitive to the occupation of seemingly uninvolved modes, turning into gates on the qubits representing them. This is the same issue that prohibits us from describing fermions on (two-dimensional) lattices in terms of bosons, which could be simulated more easily. In fact, the problems are somewhat intertwined considering that those bosonic descriptions can double as fermion-to-qubit mappings. The Jordan-Wigner transform for instance is widely used as a fermion-to-qubit mapping [24, 45, 46, 48] today, but its appearance in 1928 [19] predates the work of Feynman by half a century. The original work of Jordan and Wigner was rather meant to compare fermionic operators to the operators of (hard-core) Bosons, which are easily mapped to $(1/2)$-spins. For our purposes, the spins are immediately identified as qubits, rendering the transform a default for fermion-to-qubit mappings. However, the Jordan-Wigner transform is effectively one-dimensional and exhibits large deficits in the treatment of two-dimensional systems. In particular it fails to map a fermionic lattice model with local interactions (meaning their interaction range is bounded by a constant) to a model of locally interacting spins. In contrast to that, locally interacting spins on a lattice can be mapped to a locally interacting Boson lattice, due to the bosonic wave function not being antisymmetric [50]. While there are tricks and generalizations to circumvent the deficits of the Jordan-Wigner transform [51–54], not all of them are useful for its role in quantum simulation: there is no ultimate choice for a two-dimensional fermion-to-qubit mapping. However, there is a mapping with which locally interacting fermion and qubit lattices can

be related: the *Verstraete-Cirac transform* (VCT) [25] also known as *Auxiliary fermion mapping* [29, 55, 56], can be regarded as a manipulation of the Jordan-Wigner transform, in which additional *auxiliary* particles are added, hence the name. Other works on fermion-to-qubit mappings [27, 29, 37, 57] are based on two transforms proposed by Bravyi and Kitaev in [26]. First, there is the already mentioned Bravyi-Kitaev transform, that, compared to the Jordan-Wigner transform, exhibits an up to exponential improvement on the number of qubits that each fermionic interaction term acts on. The Bravyi-Kitaev transformation however demands a qubit connectivity that is higher than what a square lattice can offer. Second, the mapping referred to as *'Superfast simulation of fermions on a graph'* (BKSF) has the power to map local fermion lattices to local qubit lattices, but the square lattice connectivity is generally only sufficient when the underlying model is an interacting square lattice as well: to make interactions local, the mapping requires a qubit connectivity graph set by the Hamiltonian. When the given connectivity turns into a limitation, classical tools like sorting networks might be applied [58]. Most notably, there are recent attempts to incorporate swapping networks into the fermion-to-qubit mapping. With so-called fermionic swaps [26], not only qubits are swapped but also fermionic modes, in the sense that swapping operations can change the locality of their interactions in the Jordan-Wigner transform. This effectively eliminates the contribution of the fermion-to-qubit mapping to the gate cost and algorithmic depth which is then dominated by the swapping network alone [30, 59].

In this chapter, we want to abstain from swapping and sorting networks to make use of the (two-dimensional) geometric proximity of qubits inside the quantum device. In this way, the gate cost is determined by the range of interactions on the fermionic lattice and distant interactions can be simulated in parallel. For this purpose, we define two-dimensional (nonperturbative) fermion-to-qubit mappings that generalize the Jordan-Wigner transform on the square lattice. We here not only demand that local Hamiltonians of fermions are mapped to local qubit Hamiltonians but want to go beyond nearest neighbor interactions. The exchange interaction between two (distant) modes should involve only the two qubits that these modes correspond to, and some chain of qubits that connects them geometrically. This means that when we imagine the system as a fermion lattice with dimension $(\ell_1 \times \ell_2)$, we want an interaction term of any two modes to transform into a term acting on $O(m)$ qubits, when the modes have a Manhattan distance of m. As a consequence, we can bound the weight of the largest terms by $O(\ell_1+\ell_2)$, rather than $O(\ell_1\times\ell_2)$ as in the case of the Jordan-Wigner transform. In this way the entire simulation only considers operators acting on the shortest possible strings along adjacent qubits, fostering parallelization.

3.2 Results

In this chapter, we introduce a class of fermion-to-qubit mappings, that are two-dimensional generalizations of the Jordan-Wigner transform on a $\ell_1\times\ell_2$ lattice of fermionic sites. The *Auxiliary qubit mappings* (AQMs) are based on the (one-dimensional) Jordan-Wigner transform, concatenated with specific quantum (stabilizer) codes. Stabilizer codes, which play an important role in quantum error correction, encode a logical basis of 2^N degrees of freedom (here $N = \ell_1 \times \ell_2$) in a subspace of a larger system with $n > N$ qubits. The degrees of freedom left are constrained with so-called stabilizer conditions, which means there are $n - N$ (independent) qubit operators $\{S_i\}_i$ that stabilize this basis, i.e. in the logical subspace the expectation value of all stabilizers is one, $\langle S_i\rangle = 1$. In our case, the logical basis encoded is the one of the Jordan Wigner transform, to which $r = n - N$ auxiliary qubits have been added and constrained. The entire procedure is illustrated in Figure 3.1, where the AQM performs the transition from layer (a) to (c), effectively avoiding the nonlocal interactions on layer (b). The codes used for AQMs are planar on the square lattice, and we devise a unitary quantum circuit that switches in between the layers (b) and (c). This circuit has an algorithmic depth that scales with ℓ_1, the length of one of the lattice sides. There is no such operation for mappings found in prior works, the Verstraete-Cirac transform and Superfast simulation. To compare them with the AQMs, we modify the VCT and BKSF, rendering them planar codes with the Manhattan-distance property. The contributions of this chapter

- We introduce three types of Auxiliary Qubit Mappings, each requiring a different amount of auxiliary qubits. Our main result of this paper is the *square lattice AQM*, which uses $2N-\ell_1$ qubits in total. Note that in general, mappings with more auxiliary qubits will in some sense deal better with the second dimension, but none of the mappings generalizing the Jordan-Wigner transform has a total qubit number exceeding $2N$. However, one might be interested in

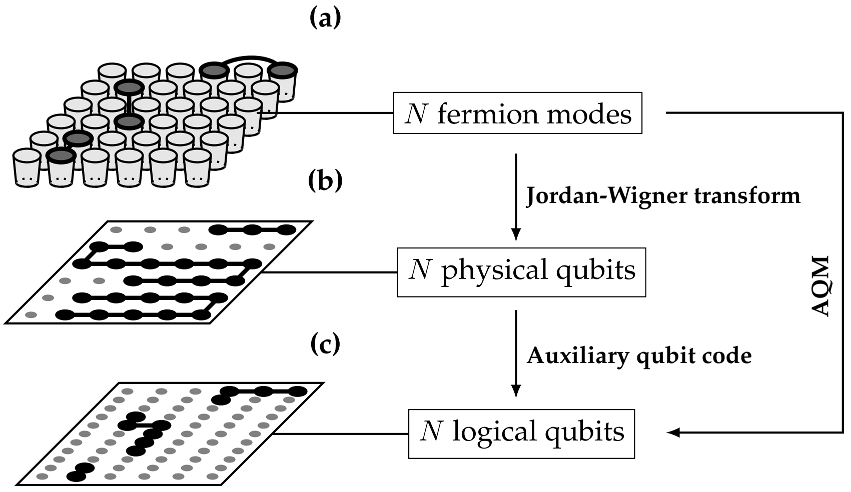


Figure 3.1. Visualizing an Auxiliary Qubit Mapping (**AQM**) as a concatenation of the Jordan-Wigner transform and a particular quantum code. The three layers represent the lattices of fermions and qubits. We have highlighted the same three exchange terms on each lattice, so their transformation can be observed. **(a)** The starting point: a fermionic lattice or two-dimensional embedding of a fermion system with $\ell_1 \times \ell_2$ modes. The three (local) interactions highlighted are brought via the Jordan-Wigner transform onto the (data) qubit layer. **(b)** The data qubit layer, in which two of the formally local interactions now assume a nonlocal form. To restore locality, we need to define a quantum code on the data qubits register and some auxiliary qubits, added to the next layer. **(c)** The final layer: a composite system of n qubits, where we have placed $n - N$ auxiliary qubits in between the data qubits. By the Auxiliary Qubit code, interactions that were local in the top layer can now be made local again. Note also that the interaction in the center of the lattice, which has involved many qubits in the middle layer, is now reduced to act on few qubits again by the Manhattan-distance property.

using fewer auxiliary qubits: this can be the case for instance when simulating lattice models, where we would like to make the physical lattice as large as possible and 'being on a fixed qubit budget' accept a trade-off between circuit depth and the number of auxiliary qubits. A qubit-economic version of this mapping would be the *sparse AQM*, which introduces the parameter $\mathcal{I}$ to regulate the trade-off. Furthermore, with adding only a few qubits we can already obtain a modified version of this mapping which has easy-to-prepare logical states and is called *E-type AQM*. A comprehensive list of all considered fermion-to-qubit mappings, that allows us to compare their properties, is compiled into Table 3.1. For all Auxiliary Qubit Mappings, we provide the initialization circuits of $O(\ell_1)$ depth.

- We demonstrate the Auxiliary Qubit Mappings on the Fermi-Hubbard model, decreasing its algorithmic depth from being linear with the number of data qubits, $O(N)$, to being constant, $O(1)$. This is an important step towards making its simulation scalable (at the expense of more qubits). Lattice models are in general not just interesting by themselves, but also test on how a fermion-to-qubit mapping deals with the second dimension, i.e. the criteria mentioned in the introduction, in a minimal fashion. We explicitly show how the mappings transform the Fermi-Hubbard model into a model of local qubit interactions on the lattice.

- We compare our work, the Auxiliary Qubit Mappings, to the Verstraete-Cirac transform [25] and the Superfast simulation [26] from the literature. As indicated above, we adjust the latter two slightly to make all three mappings comparable. Advantages and disadvantages of each mapping eventually lead us to conclude which of them to recommend for different situations.

While these contributions are covered in Sections 3.5, 3.6 and 3.7, the rest of the paper is organized as follows: in Section 3.3, we provide a more structured introduction to the layout of the quantum device and the established fermion-to-qubit mappings. We discuss criteria for a 'good' mapping in detail and that the Jordan-Wigner transform has deficits in those regards. In Section 3.4, we illustrate the effect of quantum codes, such as the ones that are the blueprint for the AQMs, on a given Hamiltonian. While the AQMs are an original idea, we cannot claim the same about their theoretical backbone: the foundations for Auxiliary Qubit codes are basically used in [60], although there the stabilizer formalism was not employed. As a consequence, one auxiliary qubit would have to be added for each term in the Hamiltonian, which is a large overhead that can be avoided by using the underlying principle to define quantum codes. We derive these codes from scratch in Section 3.9.1. Some minor contributions are provided outside the main text of this chapter. In Section 3.9.2, we study the class of tree-based mappings, to which the Bravyi-Kitaev transform belongs. The Bravyi-Kitaev transform itself does not do well with the square lattice, but we provide a general method to tailor and embed similar mappings to arbitrary two-dimensional setups. Section 3.9.3 is mostly providing details on the Verstraete-Cirac transform and Superfast simulation, but we also tackle some side issues by deriving the logical basis of both mappings.

	Jordan-Wigner (S-pattern)	Verstraete-Cirac transform	Superfast simulation	Square lattice AQM	E-type AQM	Sparse AQM
Origin	[19]	[25]	[26]	[43]	[43]	[43]
Aux. qubits	0	$\ell_1\ell_2$	$\ell_1\ell_2 - \ell_1 - \ell_2$	$\ell_1\ell_2 - \ell_1$	ℓ_2	$(\ell_2 - 1)(\frac{\ell_1 - 1}{\mathcal{I}} + 1)$
String length (general)	$O(\ell_1\ell_2)$	$O(2\ell_1 + \ell_2)$	$O(2\ell_1 + 2\ell_2)$	$O(\ell_1 + 2\ell_2)$	$O(2\ell_1 + \ell_2)$	$O(\ell_1 + 2\ell_2)$
Manhattan-distance property?	✗	✓	✓	✓	✗	approximately
String length (lattice)	$O(\ell_1)$	$O(1)$	$O(1)$	$O(1)$	$O(\ell_1)$	$O(\mathcal{I})$
Simulation time (lattice)	$O(\ell_1\ell_2)$	$O(1)$	$O(1)$	$O(1)$	$O(\ell_1\ell_2)$	$O(\mathcal{I}^2)$
+ cancellations	$O(\ell_1)$	$O(1)$	$O(1)$	$O(1)$	$O(\ell_1)$	$O(\mathcal{I})$
Restores locality?	✗	✓	✓	✓	✗	approximately

Table 3.1. All fermion-to-qubit mappings discussed in this work. We consider a $N = (\ell_1 \times \ell_2)$ square lattice block of fermionic modes, and compare the number of auxiliary qubits, or more generally the total number of qubits minus N. We also compare the scaling of the number of qubits involved in two types of Hamiltonians: generic ones, in which we expect interactions between every mode, and lattice models, with only nearest-neighbor interactions. For the former, we also ask whether long-range interactions can be mapped to operators involving qubits along a direct path (Manhattan-distance property). For the lattice models, we specify the expected algorithmic depth for simulating the entire Hamiltonian by e.g. Trotterization and whether their locality is restored after the transformation. Note that the simulation time is obtained using simulation gadgets that adhere to the square lattice connectivity of the qubits, however, we take into account that some simulation algorithms allow for partial cancellation of overlapping Pauli strings in the Hamiltonian. Note also that $\mathcal{I}$ is a parameter of the last mapping that can be chosen as some integer number: $1 \leq \mathcal{I} \leq \ell_1 - 1$. This parameter determines how well the Manhattan-distance property and locality is approximated.

3.3 Preliminaries

In this section, we describe the influence of fermion-to-qubit mappings on the algorithmic depth of quantum simulation in a setup of square-lattice qubit-connectivity. In particular, we will discuss criteria which render mappings 'good' in the sense that they allow for parallelization and low gate costs. For that purpose, we will give a theoretical description of the qubit layout and sketch the simulation algorithms. Let us start however by stating the role of fermion-to-qubit mappings for quantum simulation in general. We generally advise the reader familiar with the subject to skip ahead to Section 3.4, and if necessary use the table of notations offered in Section 3.10.

The goal of quantum simulation is to approximate the ground state and the ground-state energy of a given Hamiltonian. When the Hamiltonian acts on a space of fermions, a fermion-to-qubit mapping serves as translator between the quantum system to be simulated and the qubit system inside the quantum computer. That not only entails a correspondence of basis states, but also a transformation of the Hamiltonian. The Hamiltonian after its transformation with the mapping, is henceforward acting on the qubits inside the quantum computer. We here consider the case where the qubit system underlies architectural constraints, that we want to abstract with the following model.

Our setup is a two-dimensional quantum device that we describe with a planar graph, where each of the n vertices is a qubit. In this model, it is assumed that we can individually and simultaneously perform Pauli-rotations on every single qubit. However, entangling gates can only be applied between two qubits that share an edge in the graph. We assume that we can perform two-qubit gates individually per edge, but qubits involved in one gate cannot be part in another at the same time. Although we do not want to specify which kind of two-qubit gate is native to the quantum device, we want to assume that we can do CNOT-gates in $O(1)$ time using only a few native gates. The full qubit connectivity graph will furthermore be assumed to be a square lattice, so we can only perform entangling gates between qubits that are nearest neighbors, see Figure 3.2(a). Note that the individual connectivity graphs, that every fermion-to-qubit mapping in this chapter comes with, are subgraphs of Figure 3.2(a), such that every mapping can be embedded in the considered qubit system.

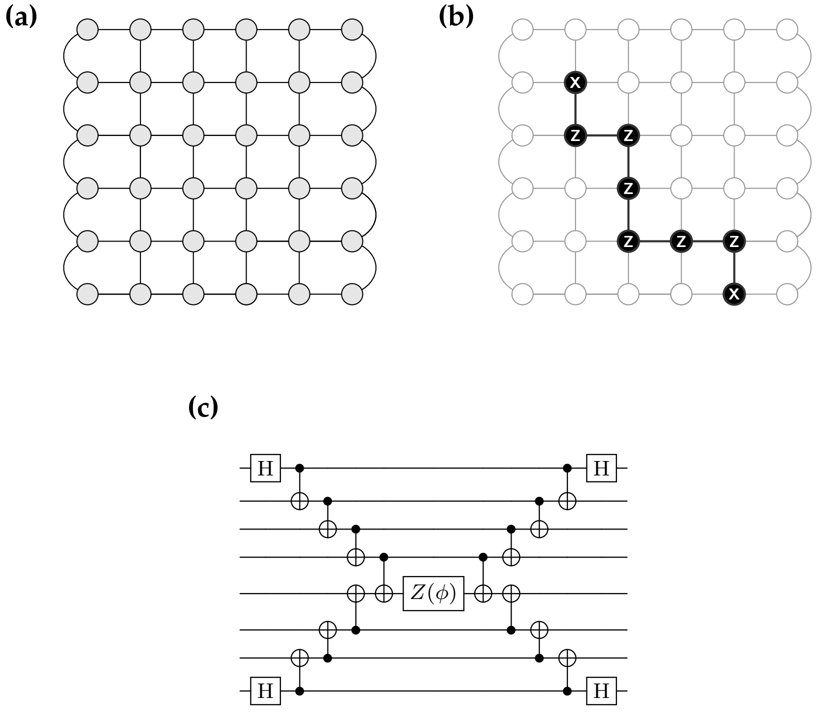


Figure 3.2. Simulation of Pauli strings in a system with limited connectivity. **(a)** Qubit connectivity graph: the vertices are qubits. Two-qubit gates can be performed only between qubits coupled by an edge. **(b)** Simulating some Pauli string $(X \otimes Z^{\otimes 6} \otimes X)$ on the quantum device: the qubits involved, and the edges along which entangling gates are performed, are highlighted. Inscriptions X, Y and Z indicate which Pauli operator acts on each qubit. **(c)** Simulating a Pauli string, here we simulate the propagator $\exp(i\,\phi\,X \otimes Z^{\otimes 6} \otimes X)$, where ϕ is an angle. The Pauli string could be the one in (b). In general, this circuit stores the parity information of the involved qubits on one of them, which is done by chains of CNOT-gates. The inscriptions X, Z and Y determine for each individual qubit whether it is in the Hadamard, computational or Y-basis in the process. Note that it does not play a role on which of the qubits the parity of the others is collected, but to optimize the simulation time, a qubit in the middle of the chain is chosen. On that qubit the phase rotation $Z(\phi) = \exp(i\,\phi\,Z)$ is performed, after which the chains are uncomputed.

3.3.1 Simulating a qubit Hamiltonian

In order to elucidate the connection between the mapping and the depth and cost of the simulation algorithms, we need to understand these algorithms better. Let us assume the fermion-to-qubit mapping transforms a Hamiltonian into the form of Pauli strings, i.e. the sum $H = \sum_h \Gamma^h \cdot h$, where $\{\Gamma^h\}$ are real coefficients associated to a Pauli string on n qubits, $h \in \{X, Y, Z, \mathbb{I}\}^{\otimes n}$. Note that we will refer to the number of qubits, that a string h acts on nontrivially, as (operator) weight and (string) length, interchangeably.
Quantum simulation algorithms have different ways to search for the ground state of H. Depending on which algorithm is used, the Pauli strings h have to be either measured, or their propagator simulated (conditionally) [5, 6]. With a propagator we mean the operator $\exp(i\,\phi\,h)$, where ϕ is an angle that typically is some function of Γ^h. Using CNOT-gates, we simulate such a propagator with the gadget like in Figure 3.2(c), where chains of these gates copy parity information across the lattice onto a single qubit, on which then a Z-rotation around the angle ϕ is performed and afterwards the CNOT-chain is uncomputed. For quantum eigensolvers, this qubit will be measured instead. Often we need the rotation to be conditional on the state of another qubit, so conventionally the Z-rotation, $Z(\phi) = \exp(i\,\phi\,Z)$, is to be replaced with a controlled rotation, $\mathbb{I} \otimes |0\rangle\langle 0| + Z(\phi) \otimes |1\rangle\langle 1|$ where the first qubit is the one that holds the parity information, and the second is the control, typically an auxiliary qubit of a phase estimation procedure. Alternatively, the quantum phase estimation algorithm can be adapted to include control qubits in the string, namely to simulate the propagator $\exp(-i\,\frac{\phi}{2}\,h \otimes Z) = \exp(-i\,\frac{\phi}{2}\,h) \otimes |0\rangle\langle 0| + \exp(i\,\frac{\phi}{2}\,h) \otimes |1\rangle\langle 1|$ instead.

For phase estimation-based algorithms, the propagator of the entire Hamiltonian, $\exp(iH\phi)$ needs to be simulated, which invokes the propagator of each string at least once (e.g. [61, 62]). Other algorithms invoke each string multiple times: Trotterization [8, 9] approximates the Hamiltonian propagator as repeating sequences of all string propagators $\exp(i\,\phi\,h)$, and in *iterative phase estimation* [23], a repeated application of $\exp(iH\phi)$ increases the accuracy of the computed energy. In general, H does not even have to be a Hamiltonian: it could also be an operator that prepares a trial state with *Givens rotations* [30] or implements a *unitary coupled-cluster* operator [63]. In any case, we will expect there to be a large number of strings in H so we would like to apply the gadgets 3.2(c) in parallel to keep the simulation shallow whenever possible. Let us coordinate the simulation of all those propagators by switching to layout diagrams like the one in Figure 3.2(b), instead of using circuit diagrams like in panel (c). This gives us an idea of all the qubits involved and how they are coupled, but leaves out certain details about for instance the specific simulation algorithm. Our ability to parallelize the simulation is determined by the fermion-to-qubit mapping, in particular in the shape of the strings that it outputs. In regard of our connectivity setup 3.2(c), we consider a fermion-to-qubit mapping as good, if it outputs Hamiltonians H with Pauli strings that are *short*, *continuous* and *non-overlapping*. We will now explain these criteria:

short - The length of a Pauli string is the number of qubits that it acts on nontrivially. While the gadget in Figure 3.2(c) implements a propagator in a number of time steps that scales linearly with the amount of qubits involved, other implementations have been conceived. As can be seen in [39, 64], the gadget can be replaced with one that performs the same operation with an up to exponential improvement in the circuit time, so at most $O(\log n)$. However, taking into account the (limited) qubit connectivity of the square lattice, we want to stick to the gadget of Figure 3.2(c). Although a time reduction can be achieved for Pauli strings acting on a nonlinearly distributed subset of qubits, we generally expect a time scaling linear in the string length. As the number of time steps is interchangeably connected to the circuit depth, we have an interest in keeping the Pauli strings as short as possible.

continuous - In general, Pauli strings in H will not only act on nearest neighbors, this means we cannot connect the qubits involved along shared edges as it is done in Figure 3.2(b). Connectivity problems are symptomatic for layouts like this, in which only nearest-neighbors are coupled. Let us assume that two qubits need to be connected in a gadget like 3.2(c), but they do not share an edge and the shortest path along edges encompasses a number of m uninvolved qubits. In order to skip these qubits, $O(m)$ additional two-qubit gates and time steps are required. In case the native two-qubit gates are either iSWAP or $\sqrt{\text{SWAP}}$, the outer qubits can be connected by a chain of SWAP gates, which costs $2m$ native gates in the former case and $4m$ in the latter. For systems with native CNOT-gates the formation SWAP gates with three CNOTs is unnecessarily expensive, so instead we amend gadgets like in Figure 3.2(c) with a construction that includes the m inner qubits in the CNOT-chains, but compensates for their contribution. We present two versions of such a compensation circuit in Figure 3.3, where the left panel shows us the gate that we would like to perform but cannot: we would like the configuration of the first qubit to be added to the last qubit by a nonlocal CNOT-gate. In the end, the circuits in the center and on the right achieve that task but render the m uninvolved qubits useless until the circuit is uncomputed. The additional cost in time and gates is $4m$, which means that it is cheaper to include a qubit in a string than to skip it. In conclusion, compensating or swapping of qubits is possible, but we would prefer to avoid the additional cost and rather deal with continuous strings.

non-overlapping - The overlap of two (or more) Pauli strings is the number of qubits in the intersection of the sets of qubits the strings act on. Two Pauli strings that are both acting nontrivially on a common subset of qubits are hard to simulate in parallel, as these qubits get parity information attached to them like in Figure 3.2(c). Unless these qubits are located at the beginning of a chain or if one string is a substring of the other, this parity would have to be corrected for. Later, we will briefly discuss the possibility of gate cancellations between similar, overlapping strings. While this has been suggested for Trotterization in [39], its impact on the approximation error is not well understood yet. Product formula approaches based on coalescing or randomization offer little or no choice in the term ordering [49, 65–67]. Thus, avoiding the need for cancellations, we ideally would like our mapping to transform all pairs of commuting fermionic operators into non-overlapping Pauli strings.

3.3.2 S-pattern Jordan-Wigner transform

Based on the insights of the previous sections, we will now review what is probably the standard fermion-to-qubit mapping [19]. In case of the Jordan-Wigner transform, the transformation matrix A can be regarded as the identity: $A = A^{-1} = \mathbb{I}$. From (2.14), we derive the number operators

$$c_j^\dagger c_j \mathrel{\hat{=}} \frac{1}{2}\left(\mathbb{I} - Z_j\right) \tag{3.1}$$

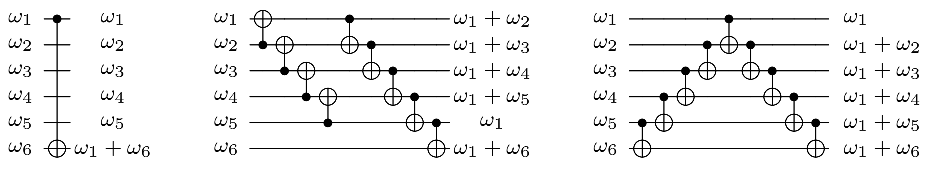


Figure 3.3. Skipping several qubits in a CNOT-chain. Here we consider the effect of the circuits on a computational basis state $(\bigotimes_i |\omega_i\rangle)$, mapping it to a state $(\bigotimes_i |\omega_i'\rangle)$. We denote the qubit values ω_i and ω_i' on the left and right side of each circuit. **Left:** The desired circuit, a CNOT-gate that adds the parity from the first qubit to the last. For connectivity reasons, this gate is not possible: we can only connect adjacent qubits. **Center/Right:** Two circuits in which the middle qubits are compensated for in order to entangle the first and last qubit. To get rid of the effect on qubits 2 - 5, the gadgets have to be partially uncomputed, but in propagators like in Figure 3.2(c), this is not necessary.

and hopping terms (for $i < j$)

$$\begin{aligned} \mathrm{h}_{ij}\, c_i^\dagger c_j + (\mathrm{h}_{ij})^*\, c_j^\dagger c_i \quad &\hat{=} \quad \frac{1}{2}\,\mathrm{Re}(\mathrm{h}_{ij}) \left(\bigotimes_{k=i+1}^{j-1} Z_k \right) (X_i \otimes X_j + Y_i \otimes Y_j) \\ &\quad + \frac{1}{2}\,\mathrm{Im}(\mathrm{h}_{ij}) \left(\bigotimes_{k=i+1}^{j-1} Z_k \right) (Y_i \otimes X_j - X_i \otimes Y_j) \ . \end{aligned} \tag{3.2}$$

While the number operator is transformed into just a constant term and a term that acts on one qubit only, the hopping terms are transformed into a string that exhibits long substrings of Z-operators, $(\bigotimes_{k=i+1}^{j-1} Z_k)$, sometimes called parity (sub-)strings. The right-hand side of (3.2), which describes an interaction of the fermionic modes i and j, translates into several strings with X- and Y-operators on the corresponding qubits of i and j, and all qubits of indices k, with $i < k < j$, are part of the parity substring. Although the parity string does us the service of connecting the qubits i and j in that way, it is also the reason that Pauli strings produced by the Jordan-Wigner transform are of length $O(N)$.
While the nature of our problem determines the Hamiltonian coefficients (such as h_{ij}) with respect to the fermionic wave functions, it is up to us to label each fermionic mode such that we minimize the appearance of long

Pauli strings in H. While problems that are intrinsically one-dimensional can be mapped to local Hamiltonians, long strings can generally not be avoided for systems in higher spatial dimensions.

The question is how to incorporate the Jordan-Wigner transform into the square lattice layout. There is a natural solution: given a $N = (\ell_1 \times \ell_2)$-matrix of qubits, we need to use only $N - 1$ edges to connect them in canonical order like beads on a string, see Figure 3.4(a). Due to the windings of the pattern on the block boundaries, we will refer to this particular way of using the Jordan-Wigner transform on a square lattice as *S-pattern Jordan-Wigner transform*. Let us now describe its properties in order to assert how good a mapping it is. The mapping produces strings that are *continuous*: although arbitrary terms (like $c_i^\dagger c_j^\dagger c_k c_l$) will in general not be transformed into continuous Pauli strings, creation/annihilation operator pairs $c_i^\dagger c_j$ will. Unfortunately the resulting Pauli-strings are neither *short* nor *non-overlapping*. As the parity strings encompass all the qubits in between i and j, the string can even span several rows, see Figure 3.4(b). This leads not just to a high gate count and algorithmic depth, but also occupies a large portion of qubits at once, effectively hindering parallelization.

Let us consider an illustrative example: if we want our quantum device to simulate a two-dimensional lattice of sites with fermionic occupation and nearest-neighbor hopping, we encounter two kinds of terms. Short ones, where the exchange between nearest-neighbors $c_i^\dagger c_{i+1}$ + h.c. yields the Pauli strings $(X_i \otimes X_{i+1} + Y_i \otimes Y_{i+1})/2$, and long ones, as the nearest-neighbor hoppings in the vertical direction will result in strings that can be seen in Figure 3.4(c). Although these are nearest-neighbor interactions, they use all qubits around the winding linking the two rows, so all vertical hopping terms between two sites in the same two rows will overlap. The S-pattern Jordan-Wigner transform thus has the property to transform operators, that are geometrically local in second quantization into nonlocal Pauli strings on the lattice. In Section 3.6, we will learn that it is those vertical hopping terms, that prevent us from simulating lattice models efficiently.

The verdict for the S-pattern Jordan-Wigner transform is that it is not good in the sense of our criteria, but good enough to serve as a foundation for better mappings. In the following, we will introduce mappings modifying the Jordan-Winger transform in using quantum codes to cancel nonlocal parity strings, which will make the resulting strings short

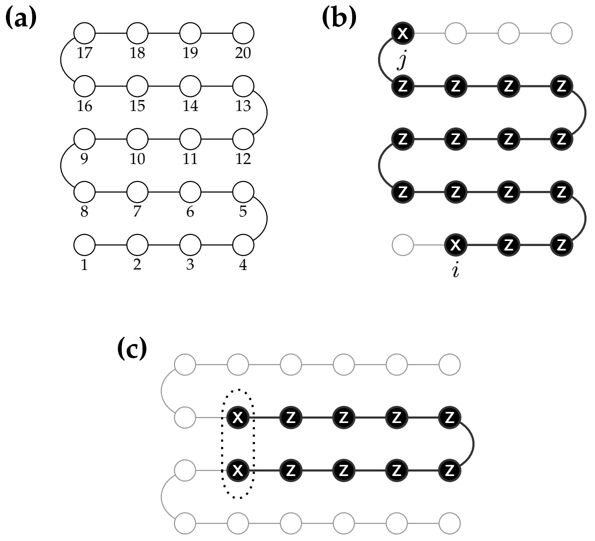


Figure 3.4. **(a)** The connectivity graph for the S-pattern Jordan-Wigner transform. **(b)** Simulating a Pauli string $(X_i \otimes Z_{i+1} \otimes \cdots \otimes Z_{j-1} \otimes X_j)$, that can be considered half of a hopping term. The string is highlighted on the device in the same way as in Figure 3.2(b). **(c)** Simulation of a Pauli string associated with a fermionic hopping between the two encircled qubits (dotted line). The hopping is in the vertical direction (diagonal to the S-pattern) which unfortunately involves gates on all qubits on the S-pattern between the two qubits.

and non-overlapping. This will lead to a certain overhead in auxiliary qubits, placed along with the original $(\ell_1 \times \ell_2)$-block of data qubits on a square lattice. In contrast to the S-pattern Jordan-Wigner transform, the mappings to follow embrace the second dimension as a useful tool.

Note that there are other alternatives to the Jordan-Wigner transform. The Bravyi-Kitaev transform [26, 27, 29, 37] is known to produce Pauli strings of weight $O(\log N)$ instead of $O(N)$. For $N > 16$ it can however be rather difficult to embed the mapping into a square lattice such that it outputs continuous strings. For a geometric interpretation of the Bravyi-Kitaev transform and related mappings we would like to refer the reader to Appendix 3.9.2.

3.4 Techniques

3.4.1 Motivation

Here we motivate the general concept of Auxiliary Qubit Mappings. The starting point will be a nonlocal Hamiltonian obtained by transformation with some linear mapping from Section 2.3. We then define quantum codes in order to restore operator locality. These codes will act on the original system extended by several 'auxiliary' qubits. The effect of such codes on the Hamiltonian will be studied.

Consider that we have an N-qubit Hamiltonian H_{dat},

$$H_{\text{dat}} \;=\; \sum_{h\in\mathcal{S}} \Gamma^h \cdot\, h_{\text{dat}}\,, \tag{3.3}$$

where $\mathcal{S}$ is the set of all Pauli strings occurring in the Hamiltonian, $\mathcal{S} \subseteq \{X, Y, Z, \mathbb{I}\}^{\otimes N}$ with all Γ^h being real, non-zero coefficients. Let us omit the qubit subscripts for now. Although we want to remain fairly general at this point, the reader can already think of (3.3) as the result of a Jordan-Wigner-transformed Hamiltonian (1.8). In general, the problem with this Hamiltonian is that $\mathcal{S}$ contains variations of Pauli strings that are either too long, discontinuous or otherwise inconvenient to us. Thus we would like to somehow replace these strings inside the Hamiltonian, even if it means that we need to add qubits to the system. Let us first consider a naïve approach which indicates the challenges of the method. We then tackle these challenges with a more sophisticated proposal. For the moment, let there be for exactly one inconvenient string $p \in \{X, Y, Z, \mathbb{I}\}^{\otimes N}$, that either appears in the Hamiltonian directly, or is the nonlocal substring of some Hamiltonian strings $\{h'\} \subset \mathcal{S}$. To bring the Hamiltonian in a convenient form, we would like to multiply every such string h' with p. Now we entangle an additional qubit to the system. Ideally, we would like to find the Pauli operator $\sigma \in \pm\{X, Y, Z\}$, acting on the added qubit, such that for every state $|\varphi\rangle$ on the original system of N qubits, there exists a state $|\widetilde{\varphi}\rangle$ on the system extended by the $(N+1)$-th qubit, on which H has the same effect as on $|\varphi\rangle$, but $(p \otimes \sigma)$ is a stabilizer:

$$(p \otimes \sigma)\,|\widetilde{\varphi}\rangle = |\widetilde{\varphi}\rangle \qquad \text{implying} \quad (p \otimes \mathbb{I})\,|\widetilde{\varphi}\rangle \;=\; \left(\mathbb{I}^{\otimes N} \otimes \sigma\right)|\widetilde{\varphi}\rangle\;. \tag{3.4}$$

If this was true, then every time p appears as a string in the Hamiltonian we could just replace it with σ, or multiply inconvenient strings $(h' \otimes \mathbb{I})$

by $(p \otimes \sigma)$ to cancel the nonlocal substrings. However, this is generally not possible: when there are terms in $\mathcal{S}$ that anticommute with p, then H will destroy the stabilizer state $|\widetilde{\varphi}\rangle$. This means that the state is altered in a way that (3.4) is no longer valid. The simulation of the adjusted Hamiltonian on such a broken stabilizer state subsequently no longer describes the correct time evolution of the underlying N-qubit system. We thus need to adjust the Hamiltonian $H \to H^{(\kappa)}$, where $H^{(\kappa)}$ generally acts on $N+1$ qubits even without having its terms multiplied by stabilizers yet. This has to be done in a way as to ensure that the time evolution of $|\widetilde{\varphi}\rangle$ according to $H^{(\kappa)}$ can be mapped back to the time evolution of $|\varphi\rangle$ according to H. At the same time we need to demand $[H^{(\kappa)}, p \otimes \sigma] = 0$ and that $(p \otimes \sigma)$ is a stabilizer like in (3.4). Only then we can use $(p \otimes \sigma)$ to cancel p inside the terms of $H^{(\kappa)}$, and so obtain a convenient Hamiltonian $\widetilde{H}$.

We now refine our approach accordingly, considering also the appearance of multiple strings p (and picking up qubit subscripts as well). In H_{dat}, we identify r Pauli strings p^i_{dat} (for $i \in [r]$) that we would like to cancel as we have done with a single string p above. Furthermore, we would like to have the option for every Hamiltonian term h_{dat} to multiply it with either several, one or none of the strings $\{p^i_{\text{dat}}\}$. This is done by repeating the above procedure for each of the r strings. To that end, we add r qubits to the system: grouping them together we introduce the r-qubit auxiliary register $\text{aux} = \{N+1, N+2, \ldots, N+r\}$. We assume that at the beginning, the aux-register is initialized in the state $|0^r\rangle = |0\rangle^{\otimes r}$. Our goal is to cancel the i-th string p^i_{dat} with a single Pauli operator on the $(N+i)$-th qubit: σ^i_{N+i}. Thus we need to find a unitary quantum circuit which entangles the aux-register with the data qubits in a certain way: it has to implement a unitary $V_{\text{aux}\,\text{dat}}$, such that for every state $|\varphi\rangle_{\text{dat}}$ (1.5), we have a state in the composite system, $|\widetilde{\varphi}\rangle_{\text{aux}\,\text{dat}}$ with

$$\begin{aligned} V_{\text{aux}\,\text{dat}}\, |\varphi\rangle_{\text{dat}} \otimes |0^r\rangle_{\text{aux}} &= |\widetilde{\varphi}\rangle_{\text{aux}\,\text{dat}} \\ \text{and} \qquad (p^i_{\text{dat}} \otimes \sigma^i_{N+i})\, |\widetilde{\varphi}\rangle_{\text{aux}\,\text{dat}} &= |\widetilde{\varphi}\rangle_{\text{aux}\,\text{dat}}\,, \end{aligned} \tag{3.5}$$

for all $i \in [r]$. To make this work even on a conceptual level, we need to demand that all p^i_{dat} commute pairwise, otherwise there cannot be a common stabilizer state of all $(p^i_{\text{dat}} \otimes \sigma^i_{N+i})$. Once the stabilizer state is obtained, we maintain it by adjusting every term of Hamiltonian (3.3) with a Pauli string on the auxiliary register. This is done in a way such that the action of the adjusted term on the enlarged system is the same as the action of the original term on the original system. The adjustments are:

$$\begin{aligned} h_{\text{dat}} &\mapsto (h_{\text{dat}} \otimes \kappa^h_{\text{aux}}) \quad \text{with} \\ V^\dagger_{\text{aux dat}} (h_{\text{dat}} \otimes \kappa^h_{\text{aux}})\, |\widetilde{\varphi}\rangle_{\text{aux dat}} &= h_{\text{dat}}\, |\varphi\rangle_{\text{dat}} \otimes |0^r\rangle_{\text{aux}} \, , \end{aligned} \tag{3.6}$$

where κ^h_{aux} is the Pauli substring on the auxiliary register that is correcting h_{dat}. Note that in case h_{dat} already commutes with all the stabilizers, κ^h_{aux} is the identity. Of course we would like the above relation to hold for every string in the Hamiltonian, $h_{\text{dat}} \in \mathcal{S}$, but as we have effectively defined a quantum code encoding the entire Hilbert space of the N data qubits, h_{dat} can be an arbitrary N-qubit Pauli string. Now by virtue of the stabilizer conditions (3.5), we can multiply the adjusted terms $(h_{\text{dat}} \otimes \kappa^h_{\text{aux}})$ by any of the operators $(p^i_{\text{dat}} \otimes \sigma^i_{N+i})$, and thus get rid of their detrimental parts. The resulting logical operators $\widetilde{h}_{\text{aux dat}}$ define a convenient (logical) Hamiltonian

$$\widetilde{H}_{\text{aux dat}} = \sum_{h \in \mathcal{S}} \Gamma^h \cdot \widetilde{h}_{\text{aux dat}} \, . \tag{3.7}$$

3.4.2 Definitions

Generally, the auxiliary qubits can be added in the computational basis to cancel strings $p^i_{\text{dat}} \in \{\mathbb{I},\, Z\}^{\otimes N}$ with Z-operators $\sigma^i_{N+i} = Z_{N+i}$. As an enhancement of the Jordan-Wigner transform, codes like this can be used to cancel nonlocal parity strings. The adjustment strings (of a term h_{dat}) κ^h_{aux} would then for all $k \in [r]$ contain X_{N+k} if h_{dat} anticommutes with p^k_{dat}. Note that the codes defined in this way (with only Z-stabilizers) have the property to map N-qubit computational basis states to states in the computational basis on n qubits, a trait that is useful for state preparation. These codes however have their limitations, as they can easily demand adjustment strings κ^h_{aux} of weight $O(r)$.

Other schemes specifically minimize the weight of κ^h_{aux} . The methods of Subaşı and Jarzynski [60] effectively define codes with auxiliary qubits in Hadamard basis that allow for an arbitrary choice of Pauli strings p^i_{dat}, as long as all r strings commute pairwise. The p-strings are subsequently replaced with X-operators, $\sigma^i_{N+i} = X_{N+i}$, and the adjustments κ^h_{aux} contain Z_{N+k} for every string p^k_{dat}, that anticommutes with h_{dat}. In [60] some concern is expressed that the operator weight might generally scale with the number of auxiliary qubits added - a key problem addressed by our work. We will in the following pick a set of strings $\{p^i_{\text{dat}}\}$ such that every term $h_{\text{dat}} \in \mathcal{S}$, resulting from any fermionic Hamiltonian, anticommutes with only a small number of stabilizers.

In Appendix 3.9.1 we give more details about these Auxiliary Qubit codes, such as their logical basis and the derivation of their stabilizers, adjustment terms as well as of the initialization unitaries $V_{\text{aux dat}}$. There are a few ways to extend the Auxiliary Qubit Mappings. In replacing the Pauli operators $\{\sigma^i_{N+i}\}$ with a set of Pauli strings $\{\gamma^i_{\text{aux}}\}$, we can even stabilize Pauli strings $\{p^i_{\text{dat}}\}$ that anticommute. In a similar vein, we can express the Verstraete-Cirac transform as a quantum code, which allows us to make modifications and to verify its operator transforms, see Appendix 3.9.3.

3.5 Auxiliary qubit mappings

3.5.1 E-type AQM

Here we present a mapping that remedies the biggest drawback of the S-pattern Jordan-Wigner transform under a moderate overhead of qubits. Given a $(\ell_1 \times \ell_2)$ block of data qubits, we are going to add ℓ_2 qubits as auxiliaries in computational basis. With this overhead, we will not manage to achieve any advantage for lattice models, but the scaling of long-range interactions (on the fermionic lattice) is improved. The following mapping will be referred to as E-type AQM. We will first illustrate its graph, along with instructions on how to initialize the stabilizer state from $|\varphi\rangle_{\text{dat}} \otimes |0^r\rangle_{\text{aux}}$. Afterwards, a discussion of the resulting Pauli strings will elucidate the advantages of the E-type AQM.
The idea of the E-type AQM is to store the parity of distinct data-qubit subsets permanently on auxiliary qubits. As we will see shortly, choosing to attach an auxiliary qubit to each of the ℓ_2 data-qubit rows is providing us with a geometric interpretation of the resulting strings. The result is shown in Figure 3.5(a). Note that two things are different between the S-pattern Jordan-Wigner transform and the E-type AQM: firstly, the connectivity graph has changed. A row of qubits is now coupled to one auxiliary qubit, and only those auxiliary qubits are coupled together, data

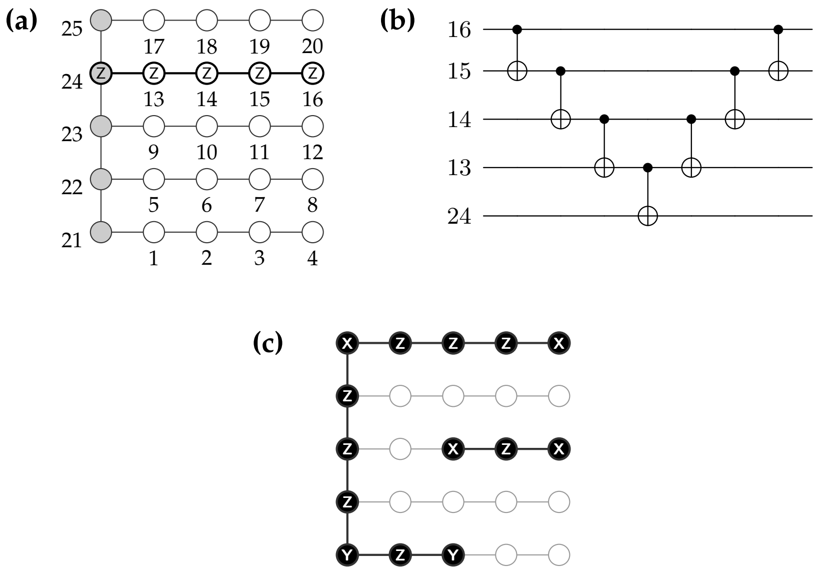


Figure 3.5. E-type AQM. **(a)** A block of (4×5) data qubits (white) enhanced with 5 auxiliary qubits (gray). A single stabilizer is highlighted in the graph. All qubits are labeled, where numberes 1-20 indicate the canonical ordering. **(b)** Initializing one of the stabilizers $(\bigotimes_{i=13}^{16} Z_i) \otimes Z_{24}$. **(c)** Simulating Pauli strings $\widetilde{h}_{\text{aux dat}}$ that are logical versions of $h_{\text{dat}} = (X \otimes Z \otimes \cdots \otimes Z \otimes X)$. The strings are highlighted as explained in Figure 3.2(b). While long strings are rerouted to skip rows, extending along the corresponding auxiliary qubits instead, shorter strings that do not switch rows can be simulated in parallel.

qubits in different rows are not coupled anymore. Although such connections between data qubits might be useful for simulating many-body terms, they are not necessarily required. Secondly, we have also changed the labeling of the qubits: the indices $i \in [\ell_1 \ell_2]$ still correspond to the indices attached to fermion operators in (2.11), but their order in the graph does no longer resemble an S-pattern of the canonical indices.
From $|\varphi\rangle_{\text{dat}} \otimes |0^r\rangle_{\text{aux}}$ the logical state $|\widetilde{\varphi}\rangle_{\text{aux dat}}$ can be initialized in $O(\ell_1)$-time and a total of $O(\ell_1 \ell_2)$ gates. Here a chain of CNOTs is used to mirror the collective parity information of an entire row of qubits on the attached auxiliary. The scaling in time is due to the fact that the preparation circuit in Figure 3.5(b), can theoretically be implemented on every row in

parallel. The stabilizers of the system are

$$\left(\bigotimes_{i \in \text{row } k} Z_i \right) \otimes Z_{N+k} , \tag{3.8}$$

for all rows $k \in [\ell_2]$ in the data qubit block. We now turn to describe the resulting Pauli strings, for which we need to discuss the adjustments κ^h_{aux}. Diagonal terms (3.2) in the Hamiltonian do not influence the stabilizer state, as well as hopping terms (3.1) between qubits in the same row. Our attention is thus focused on Pauli strings of the form $h_{\text{dat}} = (X_i \otimes Z_{i+1} \otimes \cdots \otimes Z_{j-1} \otimes X_j)$, where qubits i and j are situated in different rows k and l, where $k < l$. Those Pauli strings are subsequently adjusted by $\kappa^h_{\text{aux}} = (X_{N+k} \otimes X_{N+l})$.

In order to make these terms more convenient, we multiply the adjusted strings with the corresponding stabilizers (3.8) of rows k', for all $k \leq k' < l$. Here we discover the benefit of this mapping: wherever Pauli strings act as Z-strings on entire rows, the parity is inferred instead from the auxiliary qubits attached. This limits the length of parity substrings and so Pauli strings (originating from hopping terms) have a maximal length $2\ell_1 + \ell_2$, instead of $\ell_1\ell_2$. This is not just a benefit in time and gates, but also allows us to simulate single-row strings at the same time as long strings spanning these rows, see Figure 3.5(c).
Although we expect the E-type AQM to be useful for problems long-range interactions, it has no advantage compared to the S-pattern Jordan-Wigner transform if one considers locally-interacting lattice Hamiltonians. With only single-row Pauli strings or strings between adjacent rows, no savings in gates and algorithmic depth can be anticipated. In the following, we will define a mapping that can transform those models into local qubit-Hamiltonians.

3.5.2 Square lattice AQM

Our main result, the square lattice AQM, is a mapping that requires a square lattice connectivity graph of $\ell_1 \times (2\ell_2 - 1)$ qubits for a $(\ell_1 \times \ell_2)$ fermionic lattice. With the large amount of $\ell_1(\ell_2 - 1)$ qubits added, we make sure that the code space can be initialized in $O(\ell_1)$ time steps; a time frame that is better than linear in the total number of data qubits. In the resulting mapping, we will be able to reroute and deform Pauli

strings, such that strings originating from hopping terms have an operator weight of the order of the Manhattan distance between the two qubits on the lattice. The implication of this mapping for lattice Hamiltonians is that vertical hopping terms have a constant weight, and the algorithmic depth required to simulate such a model (after the stabilizer state is prepared) is constant, i.e. independent of the lattice dimension.
Before we start describing the mapping, we want to introduce some helpful notation concerning qubit labeling. For the sake of a geometric interpretation, we will migrate to a geometric labeling, where each qubit index denotes its coordinate on a grid. In the following, qubits in the data register will bear labels $(i,\, j) \in [\ell_1] \otimes [\ell_2]$, so each data qubit sits on integer positions of a grid and the qubit in the south-west corner of the block has coordinate $(1,\, 1)$. Beginning from that very qubit, the index of each qubit is given according to the canonical order of the S-pattern in Figure 3.4.
We will now describe the placement of the auxiliary qubits on the lattice. The idea of the square lattice AQM is to insert auxiliary qubits in between data qubits of different rows, so in between $(i,\, j)$ and $(i,\, j+1)$ into half-integer positions $(i,\, j+\frac{1}{2})$, in order to cancel the parity strings in between those qubits. However, we also want the p-strings to have (anti-)commutation relations like Majorana-pair operators. This is an integral ingredient to avoid long adjustments substrings κ^h_{aux}. To that end, we use a Hadamard-basis Auxiliary Qubit code with stabilizers

$$p_{\text{dat}}^{(i,j+\frac{1}{2})} \otimes X_{(i,j+\frac{1}{2})}\,, \tag{3.9}$$

which act on the data qubits at $(i,\, j)$ and $(i,\, j+1)$ as X- or Y-operators and as Z-operators on all other data qubits along the S-pattern in between them. The position of the auxiliary qubits and the choice of stabilizers can be seen in Figure 3.6. Note that it is unnecessary for the auxiliary qubits to be connected to each other in the horizontal direction, although it might come in handy in the process of initializing the code space. As indicated in the figure, the Pauli terms on $(i,\, j)$ and $(i,\, j+1)$ in the stabilizers of qubits $(i,\, j+\frac{1}{2})$ are different for even and odd rows numbers j. The sole reason for this decision is to render both terms of the vertical hopping terms with real coefficients (3.2) of the same weight. For every vertical connection $(i,\, j+\frac{1}{2})$, the p-substrings of the stabilizers

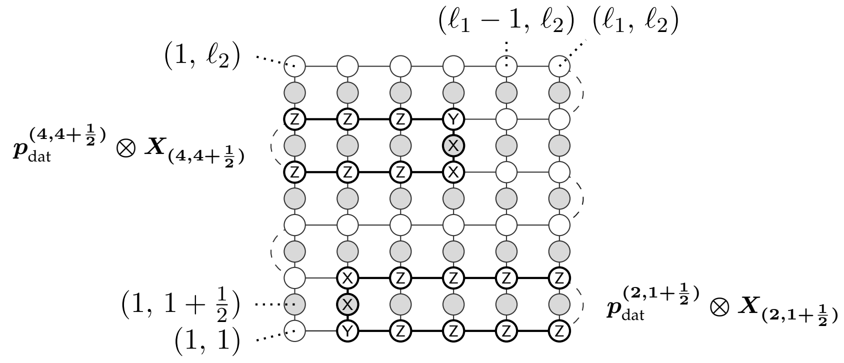

$(\ell_1-1,\,\ell_2)$
$(\ell_1,\,\ell_2)$
$(1,\,\ell_2)$
$\boldsymbol{p}_{\mathrm{dat}}^{(4,4+\frac{1}{2})}\otimes\boldsymbol{X}_{(4,4+\frac{1}{2})}$
$\boldsymbol{p}_{\mathrm{dat}}^{(2,1+\frac{1}{2})}\otimes\boldsymbol{X}_{(2,1+\frac{1}{2})}$
$(1,\,1+\frac{1}{2})$
$(1,\,1)$
Z
Y
X

Figure 3.6. Square lattice AQM, defined on a $\ell_1\times(2\ell_2-1)$ square lattice of qubits, here $\ell_1=\ell_2=6$. The gray qubits form the aux-register. Some qubits are labeled with their coordinates (dotted lines), where the auxiliary qubits generally sit on half-integer positions. The dashed lines do not couple qubits, but only indicate the windings of the S-pattern of the underlying Jordan-Wigner transform. The highlighted qubits and edges are two examples of stabilizers for odd and even rows, respectively, labeled in bold.

(3.9) are defined as:

$$p_{\mathrm{dat}}^{(i,\,j+\frac{1}{2})} = \left(\bigotimes_{k=i+1}^{\ell_1} Z_{(k,\,j)}\right)\left(\bigotimes_{l=\ell_1}^{i+1} Z_{(l,\,j+1)}\right)\otimes Y_{(i,\,j)}\otimes X_{(i,\,j+1)}\,,\quad \text{for odd } j, \tag{3.10}$$

$$= \left(\bigotimes_{k=i-1}^{1} Z_{(k,\,j)}\right)\left(\bigotimes_{l=1}^{i-1} Z_{(l,\,j+1)}\right)\otimes X_{(i,\,j)}\otimes Y_{(i,\,j+1)}\,,\quad \text{for even } j. \tag{3.11}$$

Now we are going to give instructions on how to initialize the state $|\widetilde{\varphi}\rangle$ within $O(\ell_1)$ depth, starting from a disentangled state $|\varphi\rangle_{\mathrm{dat}}\otimes|0^r\rangle_{\mathrm{aux}}$. First we apply Hadamard gates on all auxiliary qubits. In all rows with odd [even] row numbers j, we then simultaneously apply the strings $(Y_{(\ell_1,\,j)}\otimes X_{(\ell_1,\,j+1)})$ $\left[(X_{(1,\,j)}\otimes Y_{(1,\,j+1)})\right]$ conditional on the qubit at $(\ell_1,\,j+\frac{1}{2})$ $\left[(1,\,j+\frac{1}{2})\right]$. Entangling these auxiliaries is easy as the stabilizers are at the windings and therefore local, the operation can be performed in $O(1)$ time steps. We then proceed by applying the strings

$$X_{(\ell_1-s+1,\,j)}\otimes Y_{(\ell_1-s+1,\,j+1)}\otimes Y_{(\ell_1-s,\,j)}\otimes X_{(\ell_1-s+1,\,j+\frac{1}{2})}\otimes X_{(\ell_1-s,\,j+1)}$$
$$\left[Y_{(s,\,j)}\otimes X_{(s,\,j+1)}\otimes X_{(s+1,\,j)}\otimes X_{(s,\,j+\frac{1}{2})}\otimes Y_{(s+1,\,j+1)}\right] \tag{3.12}$$

conditionally on the qubits $(\ell_1 - s, j + \frac{1}{2})$ $[(s+1, j + \frac{1}{2})]$. We do this sequentially from $s = 1$ to $s = (\ell_1 - 1)$, which means we require $O(\ell_1)$ time steps in total. This concludes the definition $V_{\text{aux dat}}$, as can be verified considering its formal definition in Appendix 3.9.1, and where we use that (3.12) is obtained from the multiplication of a p-string with the closest stabilizer. A measurement-based approach for state preparation is discussed in Section 3.7.

We are now going to describe the logical operators of the code space defined. In Figure 3.7(a), the adjusted term $\widetilde{h}_{\text{aux dat}}$ to a string $h_{\text{dat}} = (X \otimes Z \otimes \cdots \otimes Z \otimes X)$ is presented. In Section 3.9.3.1 we will show that for Pauli strings originating from hopping terms (3.2) between two sites (i, j) and (k, l), it is sufficient to check for adjustments on only the auxiliary qubits at $(i, j \pm \frac{1}{2})$ and $(k, l \pm \frac{1}{2})$. If j and l are different rows, it follows that the string is not continuous, see Figure 3.7(a). We then choose to multiply the adjusted term with the stabilizers involving the auxiliary qubits on which we wish the string to cross rows. For vertical hoppings of lattice Hamiltonians, this choice is trivial. For arbitrary hoppings however it is not. Considering that we likely have several such terms inside one Hamiltonian, we want commuting strings not to overlap so we would deform them (by multiplying other stabilizers) to go around each other. This allows us to simulate them in parallel. In Figure 3.7, panels (b)-(d), different paths have been chosen for the logical operator $\widetilde{h}_{\text{aux dat}}$ to run along. Only deformed by the multiplication of stabilizers, all of those choices are in fact equivalent. Note that taking a direct path, the resulting strings will always be of roughly the same length, as every direct path connecting two nodes on a square lattice has the same distance: the Manhattan distance.

In the following, we will generalize this mapping to yield an AQM-version that requires fewer auxiliary qubits.

3.5.3 Sparse AQM

The sparse AQM is a modification of the square lattice AQM that allows us to make a trade-off between the number of auxiliary qubits required and the locality in the resulting strings. The latter directly influences the performance of any quantum simulation algorithm.

In the square lattice AQM, each data qubit (of the interior) has two nonlocal connections in the vertical direction. This can be regarded as

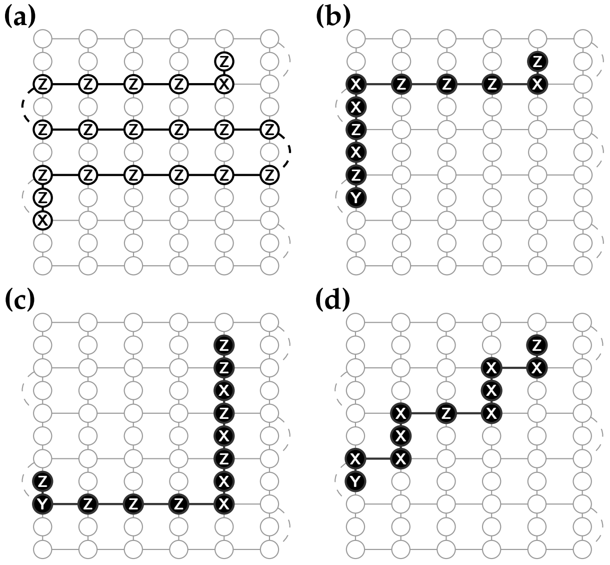


Figure 3.7. Depicted are logical representatives of the same hopping term $h_{\text{dat}} = (X \otimes Z \otimes \cdots \otimes Z \otimes X)$ spanning several rows and columns in the square lattice. The depiction of all strings follows the explanation in Figure 3.2(b). **(a)** Adjusted term $(h_{\text{dat}} \otimes \kappa^{h}_{\text{aux}})$, not yet multiplied with any stabilizer. Note that this string is not connected on the lattice, and the windings on which the string is disconnected are highlighted. **(b)-(d)** Pauli strings $\widetilde{h}_{\text{aux dat}}$ that are equivalent to $(h_{\text{dat}} \otimes \kappa^{h}_{\text{aux}})$ by multiplication with stabilizers. All those strings are continuous on the connectivity graph. The strings in (b) and (d) have the same weight (and the string in (c) is just slightly longer) which is determined by the Manhattan distance of the string endpoints.

quite wasteful, as a mapping with fewer vertical connections would work in the same way while effectively reducing the number of auxiliary qubits. Here we introduce the sparse AQM, in which vertical connections have a certain distance from each other. Let us say vertical connections are always placed $\mathcal{I}$ qubits apart. The periodicity $\mathcal{I}$ thus becomes a parameter of the mapping and is generally an integer number $\mathcal{I} \in [\ell_1 - 1]$, where the case $\mathcal{I} = 1$ reproduces the square lattice AQM. We have excluded the case in which we have only one vertical connection between every pair of rows, as it is covered by the E-type AQM already. For convenience let us say that $(\ell_1 - 1)/\mathcal{I}$ is an integer such that we can place vertical connections at the right and left boundary of the grid without spacing unequally. The connectivity graph that puts auxiliary qubits on half integer positions along $\mathcal{I}$-spaced columns can be seen in Figure 3.8(a), along with the typical stabilizers. In this mapping the auxiliary register holds $r = (\ell_2 - 1) \cdot (\frac{\ell_1 - 1}{\mathcal{I}} + 1)$ qubits, which is somewhere in between the square lattice and E-type AQM. For the initialization circuit, $V_{\text{aux dat}}$, the sequence (3.12) has to be changed into applying the strings

$$\begin{gathered}\left(X_{(\ell_1 - s + \mathcal{I},\, j+\frac{1}{2})} \otimes p_{\text{dat}}^{(\ell_1 - s + \mathcal{I},\, j+\frac{1}{2})}\right) \cdot p_{\text{dat}}^{(\ell_1 - s,\, j+\frac{1}{2})} \\ \left[\left(X_{(s+1-\mathcal{I},\, j+\frac{1}{2})} \otimes p_{\text{dat}}^{(s+1-\mathcal{I},\, j+\frac{1}{2})}\right) \cdot p_{\text{dat}}^{(s+1,\, j+\frac{1}{2})}\right]\end{gathered} \tag{3.13}$$

conditionally on qubits $(\ell_1 - s,\, j + \frac{1}{2})$ $[(s + 1,\, j + \frac{1}{2})]$
for $s = \mathcal{I},\, 2\mathcal{I},\, 3\mathcal{I},\, \dots,\, \ell_1 - 1$. All those strings in the sequence are of weight $O(\mathcal{I})$, but there are just $(\ell_1 - 1)/\mathcal{I}$ of them, which brings the depth of the entire circuit to $O(\ell_1)$.
Figure 3.8(b) shows some output strings of this mapping. While crossing rows works like in the square lattice AQM, the sparsity of vertical connections makes for a more limited choice on where the strings can run along. As a consequence, hopping terms between modes with a horizontal distance smaller than $\mathcal{I}$ will transform into strings like in the E-type mapping. The effect of sparsity on simulations of a lattice model is discussed in the following section.

Note that we have made two arbitrary design choices for the connectivity graph of this mapping: firstly, we have chosen for the auxiliary qubits to be situated in between rows of data qubits. In order to fit this mapping to a compact square lattice, we can take the auxiliary qubits from in between the rows and insert them into the rows, so e.g. take them from $(i,\, j+\frac{1}{2})$ and insert them at $(i+\frac{1}{2},\, j)$. Then, the auxiliaries have to be connected to the data qubits $(i,\, j)$ and $(i+1,\, j)$, as well as the auxiliary qubits at $(i+\frac{1}{2},\, j\pm 1)$. In the end, no qubits will be in the spaces between rows - this makes the array more dense and we can map it to a square lattice, but also requires us to skip auxiliary qubits in some horizontal hopping strings. Secondly, we have decided to place auxiliary qubits inside the same column of every other vertical connection. Alternatively, the vertical connections could be arranged in a brickwork pattern in order to minimize the weight of the adjustments κ^h_{aux}, but then vertical connections along a straight line are no longer possible.

3.6 Example: Fermi-Hubbard lattice model

3.6.1 Second quantization and Jordan-Wigner transform

Here we demonstrate the use of AQMs on the Fermi-Hubbard model. In this model, we describe spin-$\frac{1}{2}$ fermions hopping on a square lattice, with a repulsion term whenever spin-up and -down particles are present on the same site. In the following, we will describe the Hamiltonian in both, second quantization and in terms of Pauli strings after Jordan-Wigner transform. Investigating the shortcomings of this mapping with respect to circuit depth will be the motivation for the application of AQMs in the next step. Let us consider an $(L \times L)$-site square lattice of spatial sites populated by spin-(1/2) fermions: as every such site hosts a spin-up and -down mode, a total of $N = 2L^2$ qubits are minimally required. For convenience, the spin-up and -down modes of the fermionic site with the physical location $(x,\, y)$ shall be placed at the coordinates $(2x,\, y)$ and $(2x-1,\, y)$ in the two-dimensional embedding. This means the spin-partners are horizontal neighbors, which is advantageous for the Jordan-Wigner transform (and square lattice AQM). The Fermi-Hubbard Hamil-

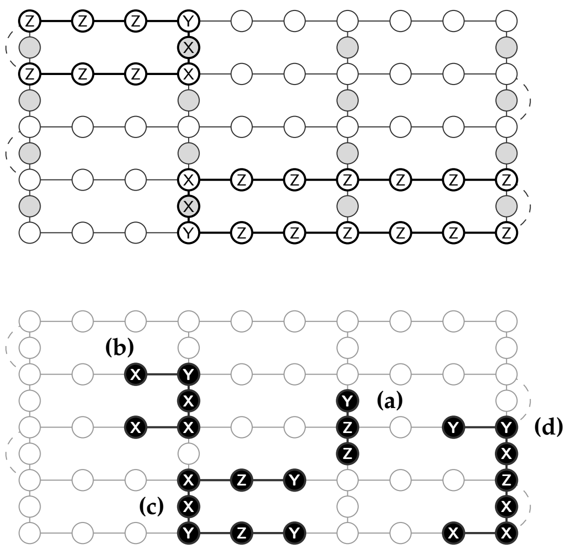


Figure 3.8. Sparse AQM with a periodicity of three ($\mathcal{I} = 3$). **Top:** Structure and stabilizers. The gray qubits are auxiliaries, placed sparsely on half-integer positions, connecting different rows. We depict one of the stabilizers in an odd and an even row, respectively. **Bottom:** Logical equivalents $\widetilde{h}_{\text{aux dat}}$ of various strings $h_{\text{dat}} = (X \otimes Z \otimes \cdots \otimes Z \otimes X)$, that originate from vertical hopping terms. **(a)** A vertical hopping along a vertical connection. The mapping yields the same $(Z \otimes Z \otimes Y)$-string as we would expect from the square lattice AQM. **(b)** The string is connecting $(3,\, 3)$ and $(3,\, 4)$. This example shows the virtue of the sparse AQM: the parity string takes a shortcut along the closest vertical connection. **(c)** Here we connect the qubits on $(6,\, 1)$ and $(6,\, 2)$ from the other direction: over the vertical connection between $(4,\, 1)$ and $(4,\, 2)$. **(d)** A next-nearest-neighbor vertical hopping term between $(9,\, 1)$ and $(9,\, 3)$.

tonian is defined as

$$\begin{aligned}
&\overbrace{\sum_{(i,j)} \left(t^{\leftrightarrow}_{ij} \; c^{\dagger}_{(i,\,j)} c_{(i+2,\,j)} + \text{h.c.} \right)}^{\text{horizontal hoppings}} + \overbrace{\sum_{(i,\,j)} \left(t^{\updownarrow}_{ij} \; c^{\dagger}_{(i,\,j)} c_{(i,\,j+1)} + \text{h.c.} \right)}^{\text{vertical hoppings}} \\
&+ \underbrace{\sum_{(i,\,j)} \epsilon_{ij} \; c^{\dagger}_{(i,\,j)} c_{(i,\,j)}}_{\text{on-site detunings}} + \underbrace{\sum_{(2i,\,j)} U_{ij} \; c^{\dagger}_{(2i,\,j)} c_{(2i,\,j)} c^{\dagger}_{(2i-1,\,j)} c_{(2i-1,\,j)}}_{\text{Hubbard interactions}} \,,
\end{aligned} \tag{3.14}$$

where $t^{\leftrightarrow}_{ij}$, $t^{\updownarrow}_{ij}$, ϵ_{ij} and U_{ij} are real parameters. In this particular example sums run over all possible coordinates $(i,\, j)$, $(2i,\, j)$ respectively, but implement open boundary conditions. With an S-pattern Jordan-Wigner transform, the Hamiltonian can now be mapped onto an $(2L \times L)$ square lattice of qubits:

$$
\begin{aligned}
H = &\sum_{(i,\,j)} \frac{t^{\leftrightarrow}_{ij}}{2} \left(X_{(i,\,j)} \otimes Z_{(i+1,\,j)} \otimes X_{(i+2,\,j)} + Y_{(i,\,j)} \otimes Z_{(i+1,\,j)} \otimes Y_{(i+2,\,j)} \right) \\
&+ \sum_{(i,\,j),\,\text{odd}\,j} \frac{t^{\updownarrow}_{ij}}{2} \left(\bigotimes_{k=i+1}^{2L} Z_{(k,\,j)} \right) \left(\bigotimes_{l=2L}^{i+1} Z_{(l,\,j+1)} \right) \left(X_{(i,\,j)} \otimes X_{(i,\,j+1)} + Y_{(i,\,j)} \otimes Y_{(i,\,j+1)} \right) \\
&+ \sum_{(i,\,j),\,\text{even}\,j} \frac{t^{\updownarrow}_{ij}}{2} \left(\bigotimes_{k=i-1}^{1} Z_{(k,\,j)} \right) \left(\bigotimes_{l=1}^{i-1} Z_{(l,\,j+1)} \right) \left(X_{(i,\,j)} \otimes X_{(i,\,j+1)} + Y_{(i,\,j)} \otimes Y_{(i,\,j+1)} \right) \\
&+ \sum_{(i,\,j)} \frac{\epsilon_{ij}}{2} \left(\mathbb{I} - Z_{(i,\,j)} \right) + \sum_{(2i,\,j)} \frac{U_{ij}}{4} \left(\mathbb{I} - Z_{(2i,\,j)} \right) \left(\mathbb{I} - Z_{(2i-1,\,j)} \right) . \qquad (3.15)
\end{aligned}
$$

Let us discuss the terms of this Hamiltonian, and finally arrive at the shortcomings of the mapping applied. We note that the vertical hopping terms are different with respect to even and odd columns, due to different directions of the S-pattern. All terms but the vertical hoppings have a constant weight and can be simulated in $O(1)$ time: only the latter can assume a length of up to $4L$. Unfortunately, we have $O(L)$ terms of weight $O(L)$ per row pair. Although these strings commute, they do overlap, which means we cannot simulate them in parallel: if no cancellations are possible, then the entire algorithm has an algorithmic depth of $O(L^2)$, so it scales with the lattice area. In this case the simulation time and the gate count cannot be better than being proportional to the total number of qubits, which renders increasing lattice size expensive. If the simulation algorithm allows us to cancel substrings of consecutively simulated Pauli strings (see for instance [39]), the algorithmic depth can improve to up to $O(L)$. To achieve even better scalings, we will employ the square lattice AQM and sparse AQM on (3.14). A detailed consideration of the E-type AQM is omitted, as it does not improve upon the scaling in case of lattice models.

3.6.2 Square lattice and sparse AQM

With the square lattice AQM, the Fermi-Hubbard Hamiltonian can be simulated in constant time, neglecting the algorithmic depth necessary to initialize the code space, which is $O(L)$ or $O(1)$ depending on the exact method used. We will now describe how the square lattice AQM modifies the terms of the Hamiltonian (3.15), after which we will discuss the sparse AQM in that regard.

We now use the square lattice AQM to render the vertical hopping terms local: after adjusting each term of (3.15) by $h_{\text{dat}} \to h_{\text{dat}} \otimes \kappa^h_{\text{aux}}$, the multiplication of adjusted hopping terms between $(i,\, j)$ and $(i,\, j+1)$ with stabilizers $(p^{(i,\, j+\frac{1}{2})}_{\text{dat}} \otimes X_{(i,\, j+\frac{1}{2})})$ is resulting in local operators of weight 3. While the hopping terms in (3.14) only have real coefficients, the operator weight of more general vertical hopping terms varies, but remains 3 on average. For complex hopping amplitudes $t^{\updownarrow}_{ij}$, we find

$$\begin{aligned} t^{\updownarrow}_{ij}\; c^{\dagger}_{(i,\, j)} c_{(i,\, j+1)} + (t^{\updownarrow}_{ij})^{*}\; c^{\dagger}_{(i,\, j+1)} c_{(i,\, j)} \quad &\hat{=} \\ \frac{(-1)^j}{2}\, \mathrm{Re}(t^{\updownarrow}_{ij}) &\left(Z_{(i,\, j-\frac{1}{2})} \otimes Z_{(i,\, j)} \otimes Y_{(i,\, j+\frac{1}{2})} \right) \\ - \frac{(-1)^j}{2}\, \mathrm{Re}(t^{\updownarrow}_{ij}) &\left(Y_{(i,\, j+\frac{1}{2})} \otimes Z_{(i,\, j+1)} \otimes Z_{(i,\, j+\frac{3}{2})} \right) \\ + \frac{(-1)^j}{2}\, \mathrm{Im}(t^{\updownarrow}_{ij}) &\left(Z_{(i,\, j-\frac{1}{2})} \otimes Z_{(i,\, j)} \otimes X_{(i,\, j+\frac{1}{2})} \otimes Z_{(i,\, j+1)} \otimes Z_{(i,\, j+\frac{3}{2})} \right) \\ - \frac{(-1)^j}{2}\, \mathrm{Im}(t^{\updownarrow}_{ij}) &\;\; X_{(i,\, j+\frac{1}{2})} \, . \end{aligned} \tag{3.16}$$

The improvements that we make on vertical terms come at the cost of the adjustments κ^h_{aux} to other terms in (3.15). However, as already mentioned, the structure of the strings $\{p^i_{\text{dat}}\}$ guarantees to keep those other terms local. For horizontal hopping terms that are (like the vertical strings) of the form $h_{\text{dat}} = (\mathrm{A}_i \otimes Z_{i+1} \otimes \ldots \otimes Z_{j-1} \otimes \mathrm{B}_j)$, with $\mathrm{A}, \mathrm{B} \in \{X,\, Y\}$, the substrings κ^h_{aux} invoke Z-operators at the end of the strings which makes for an additional weight of 2. On the other hand, if $\mathrm{A}, \mathrm{B} = Z$, κ^h_{aux} features Z-operators along the entire string. This means that while single Z-operators are in this way adjusted to $Z_{(i,\, j)} \mapsto Z_{(i,\, j-\frac{1}{2})} \otimes Z_{(i,\, j)} \otimes Z_{(i,\, j+\frac{1}{2})}$, the two-qubit Hubbard terms gain 4 qubits worth of weight.
With the square lattice AQM, we have thus managed to reduce the weight of every term to a constant independent of the system size. A list of relevant terms, that compares Jordan-Wigner and square lattice AQM can be found in Tables 3.2 and 3.3. Having achieved locality of every Hamiltonian term, we can trotterize $\widetilde{H}_{\text{aux}\,\text{dat}}$ by for instance applying all horizontal hopping terms in $O(1)$ time, then continue with a time slice in

which we simulate all vertical hoppings, follow-up with all on-site interactions and Hubbard terms, and so on. Alternatively, one may apply Hamiltonian simulation strategies to simulate patches of the lattice more accurately and then interweave these patches with the HHKL algorithm, [68].

With the square lattice AQM, we have made the simulation scalable in terms of algorithmic depth and gate count. The requirement on the qubit number has however almost doubled. In order to be more economic with the number of auxiliary qubits, we consider the sparse AQM, which will help us to maximize the size of the simulated lattice on a fixed qubit budget. Placing vertical connections $\mathcal{I}$ qubits apart, the required number of auxiliary qubits is $r = \left(\frac{2L^2-2L+1}{\mathcal{I}} + L - 1\right)$. The weight of vertical hopping strings now largely depend upon their distance to the next vertical connection: let us say there is a vertical connection across $(i,\, j+\frac{1}{2})$, then the vertical hoppings between $(i,\, j)$ and $(i,\, j+1)$ are of (constant) weight 3, like in the square lattice AQM, while the vertical hoppings of modes to their left and right rather resemble the strings of E-type AQM. The worst case is certainly met for vertical hoppings in the middle of two vertical connections, so between $(i \pm \frac{1}{2}\mathcal{I},\, j)$ and $(i \pm \frac{1}{2}\mathcal{I},\, j+1)$. Thus per vertical connection, there are $O(\mathcal{I})$ strings of weight $O(\mathcal{I})$ overlapping with one another. The simulation time is thus $O(\mathcal{I})$ if we allow cancellations and $O(\mathcal{I}^2)$ in the general case.

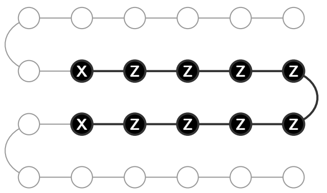
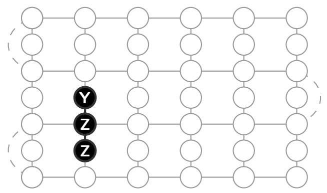
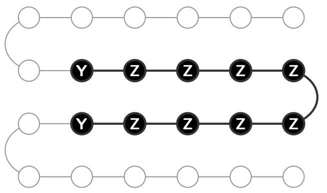
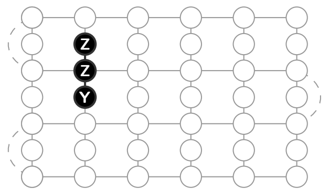
Jordan-Wigner transform		**Square lattice AQM**
		
$\left(\bigotimes_{k=i+1}^{2L} Z_{(k,j)}\right)\left(\bigotimes_{l=2L}^{i+1} Z_{(l,j)}\right) \otimes X_{(i,j)} \otimes X_{(i,j+1)}$	$\mapsto$	$-\left(Z_{(i,j-\frac{1}{2})} \otimes Z_{(i,j)} \otimes Y_{(i,j+\frac{1}{2})}\right)$
		
$\left(\bigotimes_{k=i+1}^{2L} Z_{(k,j)}\right)\left(\bigotimes_{l=2L}^{i+1} Z_{(l,j)}\right) \otimes Y_{(i,j)} \otimes Y_{(i,j+1)}$	$\mapsto$	$Y_{(i,j+\frac{1}{2})} \otimes Z_{(i,j+1)} \otimes Z_{(i,j+\frac{3}{2})}$

Table 3.2. Comparing the Jordan-Wigner transform (3.15) to square lattice AQM when applied to the Hubbard model (3.15). In this table we present vertical hopping terms – the strings of which the nonlocal part is canceled. We generally compare Jordan-Wigner strings, h_{dat} (left), to their logical equivalents $\widetilde{h}_{\text{aux dat}}$ (right) in the AQM. The strings are depicted geometrically (following the explanation of Figure 3.2(b)) and symbolically (below the drawings). Note that we display hoppings between odd rows j and even rows $j+1$ only. For j even, the two $\widetilde{h}_{\text{aux dat}}$-terms are exchanged.

3.6.3 VCT and BKSF

The Fermi-Hubbard model can also be made local by the Verstraete-Cirac transform or Superfast simulation. In this section, we will compare the weights of Pauli strings appearing in those cases to the strings resulting from transforming the Hubbard model with the square lattice AQM. We have compiled a list of the operator weights in Table 3.4, and the interested reader may find a visual representation of the strings from

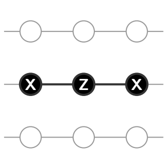
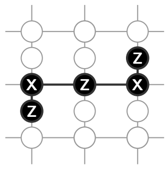
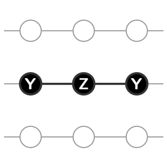
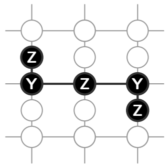
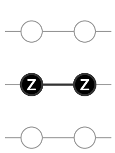
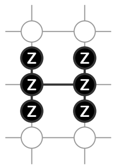
Jordan-Wigner transform		**Square lattice AQM**
		
$X_{(i,j)} \otimes Z_{(i+1,j)} \otimes X_{(i+2,j)}$	$\mapsto$	$Z_{(i,j-\frac{1}{2})} \otimes X_{(i,j)} \otimes Z_{(i+1,j)} \otimes X_{(i+2,j)} \otimes Z_{(i+2,j+\frac{1}{2})}$
		
$Y_{(i,j)} \otimes Z_{(i+1,j)} \otimes Y_{(i+2,j)}$	$\mapsto$	$Z_{(i+2,j-\frac{1}{2})} \otimes X_{(i,j)} \otimes Z_{(i+1,j)} \otimes X_{(i+2,j)} \otimes Z_{(i,j+\frac{1}{2})}$
		
$Z_{(i,j)} \otimes Z_{(i+1\,j)}$	$\mapsto$	$\bigotimes_{k\in\{0,1\}} \Big(Z_{(i+k,j-\frac{1}{2})} \otimes Z_{(i+k,j)} \otimes Z_{(i+k,j+\frac{1}{2})} \Big)$

Table 3.3. Comparing the Jordan-Wigner transform (3.15) to square lattice AQM when applied to the Hubbard model (3.15). In this table, we present the horizontal hopping and Hubbard terms – all the Pauli strings that gain in weight. However, the addition in weight is constant and local, as shown in 3.9.3. We generally compare Jordan-Wigner strings, h_{dat} (left), to their logical equivalents $\widetilde{h}_{\text{aux dat}}$ (right) in the AQM. The strings are depicted geometrically (following the explanation of Figure 3.2(b)) and symbolically (below the drawings). Not on display are the on-site terms and single-qubit contributions from Hubbard interactions, $Z_{(i,j)}$, which are adjusted into $(Z_{(i,j-\frac{1}{2})} \otimes Z_{(i,j)} \otimes Z_{(i,j+\frac{1}{2})})$.

BKSF and VCT in Appendix 3.9.3. Let us briefly discuss how the weights of the terms come to be. The VCT and AQM are quite similar in the sense that both concatenate the Jordan-Wigner transform with a quantum code. However, the data-qubit substrings of the VCT stabilizers just consist of Z-strings, which has two consequences: firstly, the stabilizers commute with diagonal terms like on-site detunings and Hubbard interactions, leaving them unadjusted and without any gain of weight. With this feature, the VCT distinguished itself from the other mapping in producing strings of the lowest weight. Secondly, while in the AQM a hopping string would just be adjusted on its end points, adjustments have to be made all along the strings in the VCT: fortunately, the auxiliary-qubit substring of the VCT stabilizers cancel these adjustments, causing this mapping to have shorter strings in the vertical direction (see Section 3.7). We thus place spin-up and -down modes of the same spatial site vertically adjacent, like we have placed them horizontally adjacent in the AQM. This leads to the weights of horizontal and vertical hoppings to be interchanged between VCT and AQM (on average). The stabilizers of both mappings can be made local with a weight of 6 (and weight-3 stabilizers at the boundaries), which is also the weight of stabilizers in the BKSF. The BKSF, defined on the least amount of qubits, has surprisingly the longest strings. The reason for this is that logical Z-operators have weight 4 - a consequence of the square lattice connectivity. With this, the BKSF has also the largest variety of weights in hopping strings, while in the VCT, there is no variety at all among strings in the same direction. While the VCT appears to be the favorable option when comparing string lengths (followed by the AQM), it also uses the most qubits, as becomes apparent in Appendix 3.9.3.

3.7 Comparison of AQM, VCT and BKSF

In this section, we will compare the Auxiliary Qubit Mapping, Superfast simulation and Verstraete-Cirac transform. Not only can the latter two be used to simulate the Hubbard model with local interactions, but we can also give them the Manhattan-distance property to align them with our notions of a good mapping for square lattices of qubits. This is done in Appendix 3.9.3. The reader completely unfamiliar with those mappings may also find an introduction reviewing the original proposals [25, 26]. Let us here compare AQM, VCT and BKSF regarding state

	Square lattice AQM	VCT	BKSF
Stabilizer (interior)	6	6	6
Vertical hoppings $XX \mid YY \mid XY \mid YX$	$3 \vert 3 \vert 5 \vert 1$	$5 \vert 5 \vert 5 \vert 5$	$2 \vert 6 \vert 5 \vert 4$
Horizontal hoppings $XX \mid YY \mid XY \mid YX$	$5 \vert 5 \vert 5 \vert 5$	$3 \vert 3 \vert 3 \vert 3$	$8 \vert 4 \vert 5 \vert 7$
Two-qubit Hubbard terms	6	2	$6+2$
On-site terms	3	1	4

Table 3.4. String lengths of the Fermi-Hubbard model transformed by all three mappings. We compare the weight of the Pauli strings, that originate from the square lattice AQM, the Verstraete-Cirac transform and the Superfast simulation. For hopping terms, we consider the strings $h_{\text{dat}} = (\mathrm{A}_i \otimes Z_{i+1} \otimes \cdots \otimes Z_{j-1} \otimes \mathrm{B}_j)$, with all variations of $\mathrm{A}, \mathrm{B} \in \{X, Y\}$. For vertical hoppings (in the AQM) we fix the case of j being in an even row. Two-qubit Hubbard terms are of the form $h_{\text{dat}} = (Z \otimes Z)$, and on-site terms are singular Z-operators. In the BKSF it is required to skip a qubit, which we penalize with an additional cost of two gates. In conclusion, the Verstraete-Cirac transform seems to exhibit the shortest strings, with the weights of the hopping terms being the same for all A_i, B_j. Regarding string lengths, the square lattice AQM is in between the Verstraete-Cirac transform and the Superfast simulation, where the latter has the longest strings and largest variations in length.

preparation, qubit requirements, Manhattan-distance property and the possibility of error mitigation. Afterwards, we can conclude and identify cases in which each mapping is advantageous.

State preparation - As we have shown, there is a unitary quantum circuit for the AQM to elevate an N-qubit state to its equivalent in the logical basis. The VCT on the other hand has a logical basis that is entangled in a more complicated way, such that we cannot find a unitary quantum circuit of the same simplicity. Although the BKSF has no clear distinction between data and auxiliary qubits, there is a set of $N-1$ qubits that is only relevant for an S-pattern and one could argue that only vertical connections add the remaining qubits and introduce stabilizers. As each connection is implemented by just one entangled qubit, we believe that there might be a unitary circuit as simple as $V_{\text{aux dat}}$. As of now, we would have to resort to syndrome measurements to initialize the code space of VCT and BKSF. By syndrome measurements, we mean the measurement and readout of a generating set of stabilizers and correct for outcomes inconsistent with the code space. While measurement and readout-times of state-of-the-art quantum devices might make this strategy challenging at present, we can at least arrange for local stabilizers such that the time overhead per measurement cycle is constant. In Figure 3.9(c)-(d) the local stabilizer tilings of VCT and BKSF are shown. A planar tiling for stabilizers of square lattice and sparse AQM follows from multiplication of adjacent stabilizer generators

$$\begin{aligned}&\left(p_{\text{dat}}^{(i,j+\frac{1}{2})}\otimes X_{(i,j+\frac{1}{2})}\right)\cdot\left(p_{\text{dat}}^{(i+1,j+\frac{1}{2})}\otimes X_{(i+1,j+\frac{1}{2})}\right)\quad\text{and}\\&\left(p_{\text{dat}}^{(i,j+\frac{1}{2})}\otimes X_{(i,j+\frac{1}{2})}\right)\cdot\left(p_{\text{dat}}^{(i+\mathcal{I},j+\frac{1}{2})}\otimes X_{(i+\mathcal{I},j+\frac{1}{2})}\right)\,,\end{aligned}\tag{3.17}$$

excluding the stabilizers at the windings, which are local already. The result is a repeating pattern of tiles with *ears* at the windings, shown in Figure 3.9(a)-(b). Note that we have implicitly used these tilings already in the respective definitions of $V_{\text{aux dat}}$. While with the unitary quantum circuit we can prepare the state on only the data qubits before encoding it into the logical basis, the same thing seems impossible with syndrome measurements. Even if the protective operations would not change the data-qubit state, there is still an ambiguity in the logical bases of VCT and AQM, that we now want to discuss. As can be seen in Appendix 3.9.1, the quantum code layer included in these mappings transform any computational basis state $|\boldsymbol{\omega}\rangle_{\text{dat}}$ into a logical basis state $\left[\prod_{i\in[r]} \frac{1}{\sqrt{2}}(\mathbb{I} + S^i_{\text{aux dat}})\right] |\boldsymbol{\omega}\rangle_{\text{dat}} \otimes |\boldsymbol{\chi}\rangle_{\text{aux}}$, where $\{S^i_{\text{aux dat}}\}_i$ is a generating set of stabilizers and $\boldsymbol{\chi} = (\chi_1, \chi_2, \dots, \chi_r)^\top \in \mathbb{Z}_2^{\otimes r}$ is a constant binary vector. While in the VCT, the set of stabilizers limit (not constrain) the choice of $\boldsymbol{\chi}$, (square lattice and sparse) AQMs are properly stabilized for all possible $\boldsymbol{\chi} \in \mathbb{Z}_2^{\otimes r}$. However, for both mappings the (signs of) adjustments made to operators h_{dat} depend on $\boldsymbol{\chi}$. For AQMs we rely on $\boldsymbol{\chi} = (0)^{\otimes r}$ for the substrings κ^h_{aux} to be free of signs. Obviously, for any basis with an unintended $\boldsymbol{\chi}$-shift, the logical Hamiltonian $\widetilde{H}_{\text{aux dat}}$ will not replicate the action of H_{dat}. As we cannot detect this $\boldsymbol{\chi}$-offset, we have to ignore it, e.g. pretend that $|\boldsymbol{\chi}\rangle = |0^r\rangle$ in AQMs: this effectively means that the state $|\widetilde{\varphi}\rangle_{\text{aux dat}}$, which is created with an unknown $\boldsymbol{\chi}$-shift in the aux-register, becomes a state $[\prod_i (p^i_{\text{dat}})^{\chi_i}]\, |\widetilde{\varphi}\rangle_{\text{aux dat}}$ without shift, a state we have not intended to prepare. To combat ambiguities in all mappings, the system has to be constrained to the correct subspace before any state preparation can happen. This means we have to measure not only the stabilizers, but also logical operators until all degrees of freedom are eliminated. Apart form the tiles, we could measure all logical Z-operators, i.e. all logical encodings of $(2c^\dagger_j c_j - 1)$. When all measurement outcomes yield '+1', we have prepared the logical zero state, $|\widetilde{0^N}\rangle_{\text{aux dat}}$. From there on, we directly prepare $|\widetilde{\varphi}\rangle_{\text{aux dat}}$ by e.g. Givens rotations [30, 69] using logical operators. This strategy appears to be the only option for measurement-based preparation of states in any mapping, although practically one will certainly want to perform only one cycle of measurements form the outcome of which the logical state and the (signs of the) stabilizers are defined. For the modest E-type AQM on the other hand, neither syndrome measurements nor unitary quantum circuits are necessary to prepare a logical state. Due to the fact that its logical basis is in the computational basis, the product state $(|0^N\rangle_{\text{dat}} \otimes |0^r\rangle_{\text{aux}})$ is in fact the logical zero state, even though the two registers are obviously not entangled. Initializing all qubits in zero at first is thus a sufficient preliminary to prepare the state $|\widetilde{\varphi}\rangle_{\text{aux dat}}$ with logical operators.

Qubit requirements - For all mappings we find the highest number of qubits they require to be $\leq 2N$, in fact only the VCT demands exactly

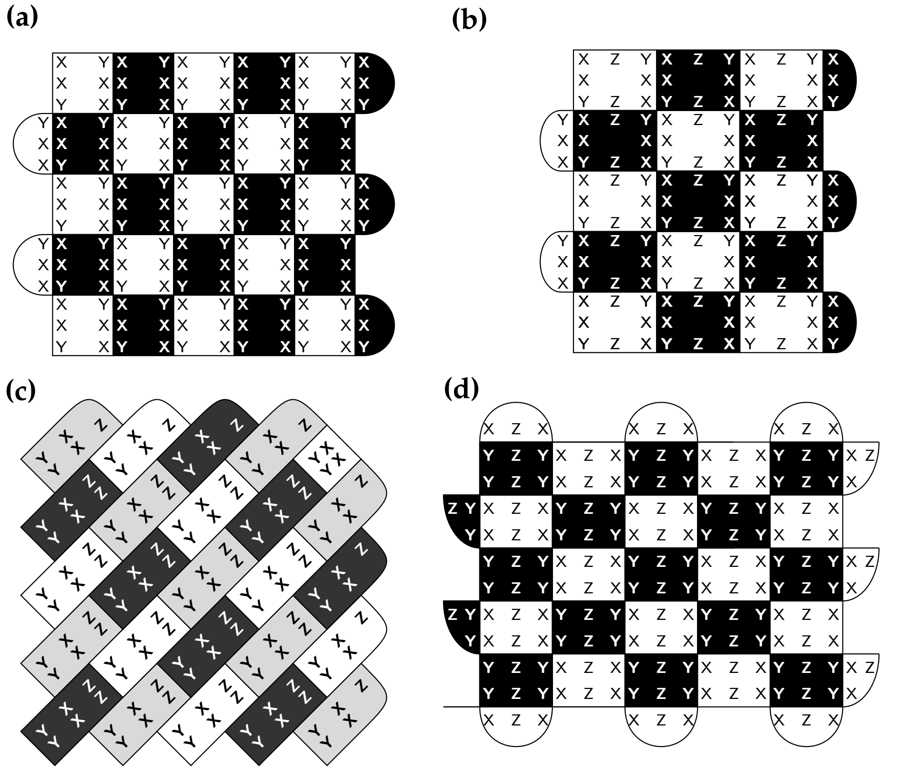


Figure 3.9. Tilings of local stabilizers for square lattice and sparse AQMs, BKSF and VCT. Every tile represents a local stabilizer involving qubits along its perimeter. Inside the tiles, X, Y and Z indicate the Pauli operators that every qubit contributes to the corresponding stabilizer. We have shaded the tiles to as a visual aid for error mitigation. **(a)** Square lattice AQM with dimensions $\ell_1 = \ell_2 = 6$. The stabilizers of all tiles are the same, except at the windings. **(b)** Sparse AQM with dimensions $\ell_1 = 7$, $\ell_2 = 6$ and $\mathcal{I} = 2$. **(c)** BKSF of a $\ell_1 = \ell_2 = 6$ fermionic lattice. The tiling is a three-colorable brickwork pattern. **(d)** VCT with dimensions $\ell_1 = \ell_2 = 6$. The stabilizer tiles are alternating in a checkerboard pattern, that resembles the rotated surface code except for the Z-operators on the data qubits.

$2N$ qubits, the square lattice AQM on the other hand requires ℓ_1 qubits less, and the BKSF requires even ℓ_2 less than the AQM. As for the AQM, we can think about reducing the amount of qubits with sparse AQMs. For the VCT such a modification is discussed in Appendix 3.9.3. As the qubits added to the VCT are generally added into the rows, its sparse version can be mapped back to a compact square lattice more easily than the AQM. In the BKSF, we can also make vertical connections more sparse, but as its layout is rotated, mapping the sparse BKSF to a compact square lattice requires changes in the connectivity graph, which will influence the continuity of resulting strings.

Manhattan-distance property - With all mappings we manage to transform long-range hopping terms of a $\ell_1 \times \ell_2$ fermionic lattice to continuous Pauli strings on a qubit lattice, that can be deformed by the multiplication of stabilizers. For all mappings, the shortest version of those strings involve a number of qubits scaling with the Manhattan distance of the fermionic modes on their lattice, but their exact weight differs from mapping to mapping - and is an interesting figure of merit. Let us say that on the fermionic lattice we have a hopping term

$$t \; c^{\dagger}_{(i,j)} c_{(i+x,\,j+y)} + t^* \; c^{\dagger}_{(i+x,\,j+y)} c_{(i,j)} \, , \tag{3.18}$$

where t and t^* is a complex coefficient and its Hermitian conjugate. Here the shortest path connecting those modes is over x modes in horizontal and y in vertical direction, the Manhattan distance is $x+y$. Transforming a string with such a distance by one of the three mappings, the connecting string is supported on roughly $O(x+y)$ qubits, but its operator weight is not going to be $x+y$ exactly. In the case of the AQM, we will have twice the number of qubits per mode in the vertical direction, which means that overcoming a vertical distance is more difficult, the string has the weight $x+2y$. In the VCT, the situation is exactly opposite and the horizontal distance is more costly to overcome due to the adjustment costs of the auxiliary modes: the operator weight of the connecting string is $2x+y$. For the BKSF, we find that horizontal and vertical paths are of equal weight, unfortunately the cost is doubled, so $2(x+y)$. Note that different versions of the BKSF exist, where the one version that yields these results is similar to the mapping in [54] - others produce strings of higher weight, for some they are even disconnected.
Note that so far we have omitted the discussion of constant weight over-

heads, that can arise at the end points of each string, and as such they are just relevant for small Manhattan distances. Around the modes labeled $(i,\, j)$ and $(i+x,\, j+y)$, BKSF and AQM can yield additional terms that matter predominantly for the local hoppings. As discussed, strings in the AQM can have one additional Z-operator around each end-mode, due to costs of the adjustments κ^h_{aux}. In the BKSF, the strings might differ by up to one logical Z-operator on each end, meaning there can be an additional cost of up to three (physical) Z-operators per end. Most notably, the VCT does not have such additional costs making it attractive for the simulation of lattice models, where $x+y$ is small.

Error mitigation - The reduction of the algorithmic depth, that all three mappings aim at, is the main tool in the reduction of noise. However, as the mappings can be regarded as stabilizer codes, it is fair to ask if they can be used for mitigating the effect of noise, as has recently been proposed on a small scale [70, 71]. Intriguingly, the AQM and VCT have local stabilizer tilings that resemble the stabilizers of surface code [17]. However, in contrast to those error correction codes, we cannot achieve topological protection against logical errors. For the planar code of the VCT to correct errors, we necessarily would need the data qubits (the qubits with Z on them in Figure 3.9(d)) to be error free, as X- and Y- errors would masquerade syndromes of errors on the auxiliary qubits. Furthermore, the code cannot detect Z-errors on the data qubits, and even increases their Z-error rate, as syndromes which are stabilizers in surface code differ by some Z-operators from the stabilizers of the VCT. A similar statement can be made for the square lattice AQM, where the auxiliary qubits would have to be perfect, and their X-error rate is increased, see Figure 3.9 (a). Using fewer auxiliary qubits, the square lattice AQM has fewer ears to mitigate errors with (as compared to Figure 3.9(d)), they could however be added with more auxiliary qubits encoding the corresponding horizontal (local) connections. Unlike the surface code, the BKSF (Figure 3.9(c)) has a three-colorable brickwork-pattern in its tiling, that theoretically allows to detect all single-Pauli errors, but like before some weight-two errors tend to masquerade themselves and go undetected when too close together. Although none of the codes allow for topological error correction, they exhibit a limited potential for error mitigation, in which one might be able to catch some errors if the rate is low enough. Whether this is feasible is left to be decided.

In conclusion, although the BKSF has the longest operators, it also requires the fewest qubits. As it is defined on a rotated square lattice, its shape might be the perfect fit for actual devices, as a patch of rotated surface code (including measurement qubits) is a rhombus. The BKSF is probably the most feasible candidate for error mitigation strategies. With its output strings having the lowest weight of all three mappings, the VCT is perhaps the most sophisticated. However, its theoretical backbone is also the most complicated – when using the VCT one would probably have to adhere to the surface-code-like structure of the original proposal. With the weight of the output strings in between the two mappings, AQMs are a compromise for the cases that demand more flexibility. The most unique feature of the AQMs is that we can just use a unitary circuit to promote a data-qubit state into its logical equivalent and if necessary even release it from the auxiliary qubits. The stabilizer state can also be manipulated during the simulation, e.g. accounting for swaps or basis transforms. The state preparation with $V_{\mathrm{aux\,dat}}$ might make this mapping even interesting for NISQ devices [4], especially for cloud-based quantum computing.

3.8 Conclusion

In this chapter, we have developed a new class of fermion-to-qubit mappings that truly generalize the Jordan-Wigner transform to two dimensions. Moreover, this class can be regarded as a quantum code layer on top of the mapping provided by the Jordan-Wigner transform, and with the unitary $V^{(\dagger)}_{\mathrm{aux\,dat}}$ we find a means to encode (decode) quantum states in the code layer. The quantum code is shown to require a certain number of auxiliary qubits that is close to N, but this number is not strict. In fact, sparse mappings with a reduced number of auxiliary qubits can achieve similar results, which might be of great practical advantage. More generally, there is a statement that we can make not just about the Auxiliary Qubit Mapping, but also the Verstraete-Cirac transform and the Bravyi-Kitaev Superfast simulation. Versions of all these transforms can be used as one-dimensional linear fermion-to-qubit mapping with N (respectively $N-1$) qubits, but at the expense of additional qubits we can pre-compute certain Pauli strings, which allows us to take shortcuts when mapping operators. This pre-computation is done when said strings are stabilized in a quantum code that entangles data qubits with the qubits added. The usage of these codes allows a quantum computer to do what was not manageable classically: the local treatment of two-dimensional fermion systems. In this way we can not only simulate fermionic lattices, but embed every fermion system on a two-dimensional layout.

We hope that future work will extend these results: we for instance have not taken into account specific limitations on either the qubit connectivity graph or the ability to perform quantum gates, which can be found in proposals for actual devices [18, 72]. It would also be interesting to incorporate the mappings into specific simulation algorithms, to see for instance how phase estimation or qubitization could deal with the planar layout.

3.9 Supplement

3.9.1 Auxiliary Qubit codes

Here we will set up the quantum codes used for the AQMs, which includes the review of the methods developed in [60]. We adapt those methods for quantum codes and contribute ideas which can be used to speed up the initialization of the logical basis.

As mentioned before, the stabilizing the Pauli strings $(p^i_{\text{dat}} \otimes \sigma^i_{N+i})$ effectively describes a quantum code: a larger Hilbert space of $n = N + r$ qubits is constrained to the dimension 2^N by r stabilizer conditions. In contrast to codes for quantum error correction, we do not want to encode information nonlocally, i.e. obtain nonlocal logical operators, but want to localize operators that were nonlocal to begin with. When characterizing a quantum error correction code, one is usually interested in the generating set of stabilizers, the logical basis states, e.g. $|\overline{0}\rangle$, $|\overline{1}\rangle$ and the logical operators, $\overline{X}$, $\overline{Z}$. In the following, we will look at the AQM equivalents of those quantities: while $\{p^i_{\text{dat}} \otimes \sigma^i_{N+i}\}_i$ is a set of stabilizer generators, the extended computational basis $V_{\text{aux dat}} |\boldsymbol{\omega}\rangle_{\text{dat}} \otimes |0^r\rangle_{\text{aux}}$ spans the logical subspace and the adjusted Pauli strings $\widetilde{h}_{\text{aux dat}}$ are its logical operators.

In the initialization of the code space via the unitary $V_{\text{aux dat}}$, the auxiliary qubits are entangled with data qubits, but not before the former are possibly rotated into some basis other than the computational basis: the basis choice of the auxiliary qubits can have consequences for other methods of state preparation and for sure determines the form of the operators σ^i_{N+i} and κ^h_{aux}. In the following, we will introduce the two logical bases, to which AQMs resort. For each of these we will outline the following points:

i. Logical basis *Basis of the $(N+r)$-qubit states $|\widetilde{\varphi}\rangle_{\text{aux dat}}$ with respect to the computational basis $|\boldsymbol{\omega}\rangle_{\text{dat}}$ of the N-qubit states $|\varphi\rangle_{\text{dat}}$.*

ii. Entangling operation *The unitary $V_{\text{aux dat}}$, for initializing the stabilizer state by quantum gates: $V_{\text{aux dat}}\,|\varphi\rangle \otimes |0^r\rangle = |\widetilde{\varphi}\rangle_{\text{aux dat}}$.*

iii. Hamiltonian adjustments *Adjustments to be made to Pauli strings, $h_{\text{dat}} \mapsto h_{\text{dat}} \otimes \kappa^h_{\text{aux}}$, and adjusted operator mappings.*

We want to deliver the last point in a two-fold way: on the one hand, we present the adjustments to a Hamiltonian in Pauli string form (3.3), where we replace every term $h_{\text{dat}} \mapsto (h_{\text{dat}} \otimes \kappa^h_{\text{aux}})$. The origin of such a Hamiltonian can be arbitrary. On the other hand we want to focus on Hamiltonians that originate from certain many-body problems of fermions. Therefore, we fuse the Hamiltonian adjustments with the linear transform, such that terms $(h_{\text{dat}} \otimes \kappa^h_{\text{aux}})$ can be obtained directly from second quantization as a redefinition of relation (2.12):

$$\begin{aligned} c_j^\dagger &\hat{=} \frac{1}{2}\left(\bigotimes_{k\in\widetilde{U}(j)} X_k\right)\left(\mathbb{I} + \bigotimes_{l\in\widetilde{F}(j)} Z_l\right)\left(\bigotimes_{m\in\widetilde{P}(j)} Z_m\right)\,, \\ c_j &\hat{=} \frac{1}{2}\left(\bigotimes_{k\in\widetilde{U}(j)} X_k\right)\left(\mathbb{I} - \bigotimes_{l\in\widetilde{F}(j)} Z_l\right)\left(\bigotimes_{m\in\widetilde{P}(j)} Z_m\right)\,. \end{aligned} \tag{3.19}$$

The redefined transform stays close to the spirit of the original in the sense that only the flip, parity and update sets are replaced by adjusted versions $\widetilde{F}(j)$, $\widetilde{P}(j)$ and $\widetilde{U}(j)$.
Apart from the two bases, we also take a look at an extension of the principle, that allows to build a stabilizer set with strings $\{p^i_{\text{dat}}\}$, that might anticommute. Interestingly, one could in this way encode all terms of a Hamiltonian into a mapping. The resulting code is perhaps most akin to the original method [60], where a new auxiliary qubit is spent for every Hamiltonian term to be multiplied with a stabilizer.

3.9.1.1 Auxiliary qubits in computational basis

With the parity strings being the detrimental substrings of the Jordan-Wigner-transformed Hamiltonians, our main goal is to cancel long strings of Z-operators. In [73], this is achieved in collecting the parity information of subsets of qubits with a circuit QED resonator. In a hardware-unspecific approach, computational basis AQMs store parity information on auxiliary qubits, which can be updated and they have never to be uncomputed.

We generally restrict computational-basis Auxiliary Qubit codes to strings $p^i_{\text{dat}} \subseteq \{\mathbb{I}, Z\}^{\otimes N}$. The p^i_{dat}-strings are here canceled with auxiliary Pauli-Z operators $\sigma^i_{N+i} = Z_{N+i}$. Let us say that the stabilizers are characterized by the $(r \times N)$ binary matrix B, such that an entry '1' in the j-th column on line i of B means that Z_j is part of p^i_{dat}:

$$p^i_{\text{dat}} \otimes \sigma^i_{N+i} \;=\; \left(\bigotimes_{j\in[N]} (Z_j)^{B_{ij}} \right) \otimes Z_{N+i} \,. \tag{3.20}$$

i. Logical basis *In the transformation to a logical state, $|\varphi\rangle_{\text{dat}} \mapsto |\widetilde{\varphi}\rangle_{\text{aux dat}}$, the computational basis is extended to*

$$|\boldsymbol{\omega}\rangle_{\text{dat}} \mapsto |\boldsymbol{\omega}\rangle_{\text{dat}} \otimes |B\boldsymbol{\omega}\rangle_{\text{aux}} \,. \tag{3.21}$$

It is easy to verify that this new basis is stabilized by (3.20) considering $Z_j \, |b\rangle_j \;=\; (-1)^b \, |b\rangle_j$, where $b \in \mathbb{Z}_2$.

ii. Entangling operation *The entangling operation can be described as a (commuting) sequence of* CNOT*-gates that depend on the matrix B. If $B_{ij} = 1$, then there is a* CNOT*-gate in $V_{\text{aux dat}}$, that, controlled on data qubit j, targets the auxiliary qubit labeled $N+i$:*

$$V_{\text{aux dat}} = \prod_{i\in[r]} \prod_{\substack{j \,\in\, [N] \\ \textit{with } B_{ij} = 1}} \textsc{CNot}\,(j \to N+i) \,. \tag{3.22}$$

The unitary $V_{\text{aux dat}}$, acting on a basis element $(|\boldsymbol{\omega}\rangle_{\text{dat}} \otimes |0^r\rangle_{\text{aux}})$ yields the extended basis of (3.21), considering that
$\textsc{CNot}(j \to k) \, |a\rangle_j \otimes |b\rangle_k \;=\; |a\rangle_j \otimes |a+b\rangle_k$, where $a,\, b \in \mathbb{Z}_2$. The entangling operation basically stores parity information of subsets of data

qubits (as defined by the rows of B) on auxiliaries. For the exact implementation of $V_{\text{aux dat}}$, (3.22) needs to be adjusted to the connectivity graph of the qubit layout. For square lattice connectivity, the above formula requires $O(rN)$ time steps in the worst case, but there is a way to improve the depth of $V_{\text{aux dat}}$: for the auxiliary qubits i and k, we can replace the circuit

$$\left[\prod_{j:B_{ij}=1} \textsc{CNot}(j \to N+i)\right]\left[\prod_{l:B_{kl}=1} \textsc{CNot}(l \to N+k)\right] \tag{3.23}$$

$$\begin{aligned} \text{by} \quad & \left[\prod_{j:B_{ij}+B_{kj}=1} \textsc{CNot}(j \to N+i)\right] \textsc{CNot}(N+k \to N+j) \\ \times \quad & \left[\prod_{l:B_{kl}=1} \textsc{CNot}(l \to N+k)\right] . \end{aligned} \tag{3.24}$$

In this (non-commuting) sequence of gates, we let the i-th auxiliary qubit inherit the parity information of the k-th auxiliary qubit by a CNOT-gate inside the aux-register. This is a useful trick when the parity information that is to be stored on these two auxiliary qubits has a large overlap in data qubits, i.e. when the vectors $\bigoplus_x(B_{ix})$ and $\bigoplus_y(B_{ky})$ have a small Hamming distance. In that case, the leftmost product contains only few CNOT-gates, as the bulk of the parity information has been inherited from the $(N+k)$-th qubit.

iii. Hamiltonian adjustments *To maintain the stabilizer state* (3.6), *we adjust a Pauli string h_{dat} on the data qubits by $h_{\text{dat}} \mapsto (h_{\text{dat}} \otimes \kappa^h_{\text{aux}})$ with*

$$\kappa^h_{\text{aux}} = \bigotimes_{m\in[r]} (X_{N+m})^{\lambda_m} , \tag{3.25}$$

where $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \dots, \lambda_r)^\top \in \mathbb{Z}_2^{\otimes r}$ is obtained by

$$\boldsymbol{\lambda} = \sum_j B\boldsymbol{u_j} \tag{3.26}$$

with $\boldsymbol{u_j}$ being the j-th unit vector of $\mathbb{Z}_2^{\otimes N}$, and the sum extending over all $j \in [N]$, for which h_{dat} acts on the qubit space as X_j or Y_j. Hamiltonian *of adjusted terms* $(h_{\text{dat}} \otimes \kappa^h_{\text{aux}})$ *as in* (3.6) *can be obtained by the redefined transforms* (3.19)*, with the same flip and parity sets,* $\widetilde{F}(j) = F(j)$ *and* $\widetilde{P}(j) = P(j)$*, but the sets* $\widetilde{U}(j)$ *defined from the columns of the matrix*

$$\left[\begin{array}{c} A \\ \hline B \end{array}\right] . \tag{3.27}$$

In case a Pauli string h_{dat} flips a data qubit, that is entangled with a qubit in the aux-register, we have to flip the latter qubit as well. In fact we need to flip all other auxiliaries to which the data qubit contributes: so if we apply the operator X_j to a basis state $|\boldsymbol{\omega}\rangle_{\text{dat}} \otimes |B\boldsymbol{\omega}\rangle_{\text{aux}}$ for $j \in [N]$, we leave the stabilized basis, unless we update the configuration of the auxiliary qubits by $B\boldsymbol{\omega} \mapsto B(\boldsymbol{\omega} + \boldsymbol{u_j})$.

Example

Let us consider a minimal example, in which the data register holds five qubits, and a sixth, an auxiliary qubit, is in the configuration $B\boldsymbol{\omega}$, where B is a (1×5) binary matrix. We consider a Hamiltonian term $h_{\text{dat}} = (X_1 \otimes Z_2 \otimes Z_3 \otimes Z_4 \otimes X_5)$. After adjusting $h_{\text{dat}} \to (h_{\text{dat}} \otimes \kappa^h_{\text{aux}})$, we have the choice to multiply with the stabilizer or not. In Table 3.5 we present the adjusted Hamiltonian before and after multiplication with the stabilizer, considering different choices of B.

B	$h_{\text{dat}} \otimes \kappa^h_{\text{aux}}$	$(h_{\text{dat}} \otimes \kappa^h_{\text{aux}}) \cdot (p^1_{\text{dat}} \otimes Z_6)$
[0 1 0 0 0]	$(X_1 \otimes Z_2 \otimes Z_3 \otimes Z_4 \otimes X_5)$	$(X_1 \otimes Z_3 \otimes Z_4 \otimes X_5 \otimes Z_6)$
[0 1 1 1 0]	$(X_1 \otimes Z_2 \otimes Z_3 \otimes Z_4 \otimes X_5)$	$(X_1 \otimes X_5 \otimes Z_6)$
[1 1 1 0 0]	$(X_1 \otimes Z_2 \otimes Z_3 \otimes Z_4 \otimes X_5 \otimes X_6)$	$-(Y_1 \otimes Z_4 \otimes X_5 \otimes Y_6)$
[1 1 1 1 1]	$(X_1 \otimes Z_2 \otimes Z_3 \otimes Z_4 \otimes X_5)$	$-(Y_1 \otimes Y_5 \otimes Z_6)$

Table 3.5. Adjusted Hamiltonian terms $\widetilde{h}_{\text{aux dat}}$ with respect to the original string $h_{\text{dat}} = (X_1 \otimes Z_2 \otimes Z_3 \otimes Z_4 \otimes X_5)$, depending on the matrix (1×5) matrix B.

3.9.1.2 Auxiliary qubits in Hadamard basis

Extending the idea of [60], we can cancel a set of arbitrary (commuting) strings $\{p^i_{\text{dat}}\}$, where $p^i_{\text{dat}} \in \{X, Y, Z, \mathbb{I}\}^{\otimes N}$, by X-operators: $\sigma^i_{N+i} = X_{N+i}$. Let us characterize the choice of the strings p^i_{dat} by three $(r \times N)$ binary matrices C^X, C^Y and C^Z. Here an entry '1' in C^s_{ji}, with $s \in \{X, Y, Z\}$, indicates that the string p^i_{dat} acts as s on the j-th qubit.

i. Logical basis *In the transformation $|\varphi\rangle_{\text{dat}} \mapsto |\widetilde{\varphi}\rangle_{\text{aux dat}}$, the computational basis is extended to*

$$|\boldsymbol{\omega}\rangle_{\text{dat}} \mapsto \left[\prod_{i\in[r]} \frac{1}{\sqrt{2}}\left(\mathbb{I} + p^i_{\text{dat}} \otimes X_{N+i}\right)\right] |\boldsymbol{\omega}\rangle_{\text{dat}} \otimes |0^r\rangle_{\text{aux}}$$
$$= \frac{1}{2^{r/2}} \sum_{\boldsymbol{\mu}\in\mathbb{Z}_2^{\otimes r}} \left[\prod_{k\in[r]} \left(p^k_{\text{dat}}\right)^{\mu_k}\right] |\boldsymbol{\omega}\rangle_{\text{dat}} \otimes |\boldsymbol{\mu}\rangle_{\text{aux}} . \tag{3.28}$$

The sum in (3.28) invoke all the possible qubit configurations $\boldsymbol{\mu} \in \mathbb{Z}_2^{\otimes r}$ with equal weight. This is a result of the auxiliary qubits being in Hadamard basis. This choice of basis becomes plausible by multiplying a basis state (3.28) with one of the stabilizers $(p^i_{\text{dat}} \otimes X_{N+i})$:

$$\left(p^i_{\text{dat}} \otimes X_{N+i}\right) \frac{1}{2^{r/2}} \sum_{\boldsymbol{\mu}\in\mathbb{Z}_2^{\otimes r}} \left[\prod_{k\in[r]} \left(p^k_{\text{dat}}\right)^{\mu_k}\right] |\boldsymbol{\omega}\rangle_{\text{dat}} \otimes |\boldsymbol{\mu}\rangle_{\text{aux}}$$
$$= \frac{1}{2^{r/2}} \sum_{\boldsymbol{\mu}\in\mathbb{Z}_2^{\otimes r}} \left[\prod_{k\in[r]} \left(p^k_{\text{dat}}\right)^{\mu_k+\delta_{ik}}\right] |\boldsymbol{\omega}\rangle_{\text{dat}} \otimes |\boldsymbol{\mu} + \boldsymbol{u_i}\rangle_{\text{aux}} . \tag{3.29}$$

If we now shift the binary vector in the sum by the i-th unit vector $\boldsymbol{u_i}$ to $\boldsymbol{\mu} \mapsto \boldsymbol{\mu} + \boldsymbol{u_i}$, the original basis element on the right-hand side of (3.28) is recovered and thus the set of Pauli strings $(p^i_{\text{dat}} \otimes X_{N+i})$ stabilizes every state $|\widetilde{\varphi}\rangle_{\text{aux dat}}$ that is in the subspace spanned by (3.28).

ii. Entangling operation *Following [60], the entangling operation can be described as*

$$V_{\text{aux dat}} = \prod_{i\in[r]} \left(|0\rangle\langle 0|_{N+i} + p^i_{\text{dat}} \otimes |1\rangle\langle 1|_{N+i}\right) \mathrm{H}_{N+i} , \tag{3.30}$$

where H_{N+i} is the Hadamard gate on the $(N+i)$-th qubit. In words, $V_{\text{aux dat}}$ can be realized by a unitary quantum circuit that first applies Hadamard gates to every auxiliary qubit, and then applies each string p^k_{dat} controlled on the k-th auxiliary qubit.

We notice that the circuit (3.30), when acting on a state $|\varphi\rangle_{\text{dat}} \otimes |0^r\rangle_{\text{aux}}$, firstly changes the basis of the auxiliary register into

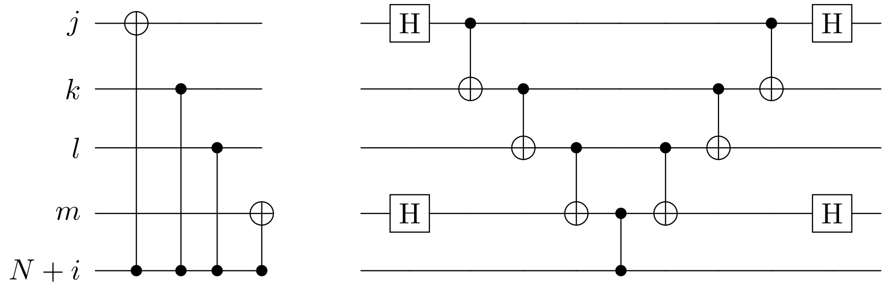


Figure 3.10. Two versions of the controlled application of the Pauli string $p^i_{\text{dat}} = (X_j \otimes Z_k \otimes Z_l \otimes X_m)$ on the i-th qubit in the auxiliary register.

$|+^r\rangle_{\text{aux}} = (\bigotimes_{i\in[r]} |+\rangle_{N+i}) = r^{-\frac{1}{2}} \sum_{\boldsymbol{\mu}\in\mathbb{Z}_2^{\otimes r}} |\boldsymbol{\mu}\rangle_{\text{aux}}$. Then the controlled application of the strings p^i_{dat} entangles auxiliary and data qubits. In principle, this can be done by CNOT, CPHASE and controlled-Y gates according to the action of a string p^i_{dat} on each data qubit, see Figure 3.10 (left). In practice, the required qubit connectivity might however not be available, such that we may resort to an implementation of the circuit as in Figure 3.10 (right). Like for the codes with computational-basis auxiliary qubits, we can here apply tricks to make $V_{\text{aux dat}}$ more shallow whenever two strings $p^i_{\text{dat}}, p^k_{\text{dat}}$ are similar to one another: after the Hadamard-gates are applied to the auxiliary qubits i and k, we can replace the circuit

$$\left(|0\rangle\langle 0|_{N+i} + p^i_{\text{dat}} \otimes |1\rangle\langle 1|_{N+i}\right) \left(|0\rangle\langle 0|_{N+k} + p^k_{\text{dat}} \otimes |1\rangle\langle 1|_{N+k}\right) \quad \text{by} \tag{3.31}$$

$$\left(|0\rangle\langle 0|_{N+i} + \left(p^i_{\text{dat}} \cdot p^k_{\text{dat}}\right) \otimes X_{N+k} \otimes |1\rangle\langle 1|_{N+i}\right) \left(|0\rangle\langle 0|_{N+k} + p^k_{\text{dat}} \otimes |1\rangle\langle 1|_{N+k}\right) , \tag{3.32}$$

which means that instead of applying the string p^i_{dat}, we conditionally apply the string that results from the operator product of p^i_{dat} with p^k_{dat}, and an X-operator on the k-th auxiliary qubit. What we use here is the fact that the $(N+k)$-th qubit is already entangled with the data qubits after the right sequence of controlled gates, such that we can use the stabilizer condition in the sequence on the left. For this to work, the order in which the two resulting strings are initialized is now fixed. A minus sign that might occur in the operator product can be reproduced by adding a Z_{N+i}, [42]. Before presenting the Hamiltonian adjustments, it is left for us to verify that the controlled applications of p^i_{dat} on $|\boldsymbol{\omega}\rangle_{\text{dat}} \otimes |+^r\rangle_{\text{aux}}$ yield the corresponding element of the extended basis (3.28). Let us consider the following reformulation of the controlled-(p^i_{dat}) terms:

$$\begin{aligned}
&\prod_{i\in[r]} \left(|0\rangle\!\langle 0|_{N+i} + p^i_{\text{dat}} \otimes |1\rangle\!\langle 1|_{N+i}\right) \\
&\quad = \prod_{i\in[r]} \left(\sum_{\mu'_i \in \mathbb{Z}_2} (p^i_{\text{dat}})^{\mu'_i} \otimes |\mu'_i\rangle\!\langle \mu'_i|_{N+i} \right) \\
&\quad = \sum_{\boldsymbol{\mu'}\in\mathbb{Z}_2^{\otimes r}} \left[\prod_{k\in[r]} \left(p^k_{\text{dat}}\right)^{\mu'_k} \right] \otimes |\boldsymbol{\mu'}\rangle\!\langle \boldsymbol{\mu'}|_{\text{aux}} \,.
\end{aligned} \tag{3.33}$$

Considering the expansion of $|+^r\rangle_{\text{aux}}$ in the computational basis, we can proceed to arrive at (3.28) by inspection.

iii. Hamiltonian adjustments *For a Pauli string h_{dat} to maintain the stabilizer state* (3.6)*, we adjust it by*

$$\kappa^h_{\text{aux}} = \bigotimes_{j\in T(h)} Z_{N+j} \,, \tag{3.34}$$

where the set $T(h) \subseteq [r]$ contains k if p^k_{dat} anticommutes with h_{dat}. As a consequence, a Hamiltonian of terms $(h_{\text{dat}} \otimes \kappa^h_{\text{aux}})$ can be obtained from second quantization using the redefined transformations (3.19)*, where the update sets are defined as before $\widetilde{U}(j) = U(j)$, but flip and parity sets $\widetilde{F}(j)$, $\widetilde{P}(j)$ are redefined by the rows of the matrices*

$$\left[A \,\middle|\, C^X + C^Y\right] \,, \quad \left[RA \,\middle|\, R(C^X + C^Y) + C^Y + C^Z\right] \,. \tag{3.35}$$

We will now show that the adjusted Pauli string $(h_{\text{dat}} \otimes \kappa^h_A)$ acts on a state $|\widetilde{\varphi}\rangle_{\text{aux dat}}$ such that after application of $V^\dagger_{\text{aux dat}}$, we recover $h_{\text{dat}}\,|\varphi\rangle_{\text{dat}} \otimes |0^r\rangle_{\text{aux}}$. We start by applying the adjusted term to the extended state. The goal is to use (anti-)commutation relations with the strings p^k_{dat} to let h_{dat} act on the data register first. It turns out that minus signs that we pick up by anticommutations are exactly canceled by sign changes originating from κ^h_A acting on the aux-register.

In general, we find if h_{dat} now anticommutes with a string p^k_{dat}, then $k \in T(h)$ such that $(h_{\text{dat}} \otimes \kappa^h_{\text{aux}})$ commutes with $(\mathbb{I} + p^k_{\text{dat}} \otimes X_{N+k})$, and we find (3.5) satisfied. For the transform (3.35), we take into account all sorts of Pauli operators that originate from parity, update and flip operators, by which we mean the strings $(\bigotimes_{m\in P(j)} Z_m)$, $(\bigotimes_{k\in U(j)} X_k)$ and

$(\bigotimes_{l\in F(j)} Z_l)$ in (2.12). If X- and Y-operators in a string p^i_{dat} anticommute with the Z-operators in the j-th flip operator, we have to counteract by adjusting it with a Z-operator on the i-th auxiliary: $(\bigotimes_{l\in F(j)} Z_l) \otimes Z_{N+i}$. The same argument holds for the parity operators, but we also add Z-operators there, stemming from anticommutations of the update operator with Z- and Y-operators in p^i_{dat}. Considering that the operators X, Y and Z appear in the strings p^i_{dat} according to the C-matrices, we can use these matrices to describe the contents of the flip and parity sets, by which we obtain (3.35).

Example

As an example we examine a 5-qubit Hamiltonian term, $h_{\text{dat}} = (X_1 \otimes Z_2 \otimes Z_3 \otimes Z_4 \otimes X_5)$. The sixth qubit is a Hadamard-basis auxiliary, used to cancel various substrings p^1_{dat}. In Table 3.6, we find the adjusted terms $(h_{\text{dat}} \otimes \kappa^h_{\text{aux}})$ and the deformed terms, $(p^1_{\text{dat}} \otimes X_6) \cdot (h_{\text{dat}} \otimes \kappa^h_{\text{aux}})$, for various choices of the stabilizer $(p^1_{\text{dat}} \otimes X_6)$.

p^1_{dat}	$(h_{\text{dat}} \otimes \kappa^h_{\text{aux}})$	$(p^1_{\text{dat}} \otimes X_6) \cdot (h_{\text{dat}} \otimes \kappa^h_{\text{aux}})$
$(Z_2 \otimes Z_3 \otimes Z_4)$	$(X_1 \otimes Z_2 \otimes Z_3 \otimes Z_4 \otimes X_5)$	$(X_1 \otimes X_5 \otimes X_6)$
$(X_1 \otimes Z_2 \otimes Z_3 \otimes X_4)$	$(X_1 \otimes Z_2 \otimes Z_3 \otimes Z_4 \otimes X_5 \otimes Z_6)$	$-(Y_4 \otimes X_5 \otimes Y_6)$
$(X_1 \otimes Z_2 \otimes Z_3 \otimes Z_4 \otimes X_5)$	$(X_1 \otimes Z_2 \otimes Z_3 \otimes Z_4 \otimes X_5)$	X_6

Table 3.6. Adjusted Hamiltonians $\widetilde{h}_{\text{aux dat}}$ to $h_{\text{dat}} = (X_1 \otimes Z_2 \otimes Z_3 \otimes Z_4 \otimes X_5)$, depending on the choice of p^1_{dat}.

3.9.1.3 Stabilizing anticommuting data-qubit strings

We present a more general quantum code based on auxiliary qubits in Hadamard basis, but in which the strings $\{p^i_{\text{dat}}\}$ do not necessarily have to commute. Using this code, an entire Hamiltonian can in principle be transformed into interactions on only the auxiliary qubits. The general idea here is to amend the scheme by the following notion: in order to counter anticommutations, we replace the (single-qubit) Pauli operators σ^i_{N+i} with Pauli strings on the auxiliary register γ^i_{aux}, such that γ^i_{aux} contains X_{N+i} as before, but for every other string p^k_{dat} with $k < i$, that anticommutes with p^i_{dat}, it contains a Z-operator, Z_{N+k}. For convenience we define the operation $\star$ as:

$$i \star k = \begin{cases} 0 & \text{if} \quad [p^i_{\text{dat}}, p^k_{\text{dat}}] \ \ = 0 \\ 1 & \text{if} \quad [p^i_{\text{dat}}, p^k_{\text{dat}}]_+ = 0 \end{cases} . \tag{3.36}$$

Using this notation, we define the stabilizers of our system as

$$p^i_{\text{dat}} \otimes \gamma^i_{\text{aux}} \ = \ p^i_{\text{dat}} \otimes \left(\bigotimes_{k \in [i-1]} (Z_{N+k})^{i \star k} \right) \otimes X_{N+i} \, , \tag{3.37}$$

since all Pauli strings $(p^i_{\text{dat}} \otimes \gamma^i_{\text{aux}})$ have to commute pairwise for all $i \in [r]$ as defined above. We will now turn to describe the mapping in the established way.

i. Extended basis *The computational basis $|\boldsymbol{\omega}\rangle_{\text{dat}}$ is extended to:*

$$|\boldsymbol{\omega}\rangle_{\text{dat}} \ \mapsto \ \frac{1}{2^{r/2}} \sum_{\boldsymbol{\mu} \in \mathbb{Z}_2^{\otimes r}} \left[(p^1_{\text{dat}})^{\mu_1} \ \cdots \ (p^r_{\text{dat}})^{\mu_r} \right] |\boldsymbol{\omega}\rangle_{\text{dat}} \otimes |\boldsymbol{\mu}\rangle_{\text{aux}} \, . \tag{3.38}$$

This basis resembles (3.28), with the subtle difference that the order of the strings p^i_{dat} matters here. When stabilizer $(p^i_{\text{dat}} \otimes \gamma^i_{\text{aux}})$ are multiplied to (3.38) from the right, the operators γ^i_{aux} cancel all minus signs from anticommutations, and flip the i-th qubit in the auxiliary register. Note that the order of the strings p^i_{dat} in (3.38) is to be taken into account when we attempt to encode $|\widetilde{\varphi}\rangle_{\text{aux dat}}$ from $|\varphi\rangle \otimes |0^r\rangle_{\text{aux}}$.

ii. Entangling operation *We pick a sequence i_1, i_2, ..., i_r that is some permutation of* 1, 2, ..., r, *in which we want to perform the entangling operation for the stabilizers $(p^{i_m}_{\text{dat}} \otimes \gamma^{i_m}_{\text{aux}})$, where the stabilizer of number i_r is taken care of first, and the one labeled i_1 last. The entangling operation associated with that sequence is*

$$V_{\text{aux dat}} \ = \ \prod_{m=1}^{r} \left(|0\rangle\langle 0|_{N+i_m} + p^{i_m}_{\text{dat}} \otimes |1\rangle\langle 1|_{N+i_m} \otimes \left[\bigotimes_{k>m} (Z_{N+i_k})^{(i_m \star i_k)\, \theta_{i_m i_k}} \right] \right) \mathrm{H}_{N+i_m} \, , \tag{3.39}$$

Note that if the we pick the original order, $i_m = m$, the circuit almost looks like (3.30), but, again, here the exact order matters. The Hamiltonian adjustments are identical to (3.34) and (3.35), as the only difference, the ordering of the strings p^i_{dat}, does not matter there: a Hamiltonian term

$\widetilde{h}_{\text{aux dat}}$ needs to pass all p^k_{dat} in (3.38), picking up all minus signs possible. We have thus obtained an auxiliary qubit mapping with completely arbitrary set of strings p^i_{dat}. If this string is a Hamiltonian term $h_{\text{dat}} = p^i_{\text{dat}}$, we can eliminate its action on the data qubits by replacing

$$h_{\text{dat}} \mapsto (h_{\text{dat}} \otimes \kappa^h_{\text{aux}}) \cdot \left(p^i_{\text{dat}} \otimes \gamma^i_{\text{aux}}\right) = X_{N+i} \otimes \left[\bigotimes_{k>i} (Z_{N+k})^{i\star k} \right] . \tag{3.40}$$

The entire Hamiltonian can in this way be pre-computed and reduced to an action on only the auxiliary register.

3.9.2 Tree-based transforms

In this section, we consider fermion-to-qubit mappings defined on tree structures for a setup with limited connectivity. This particular class of mappings is part of the mappings considered in Section 2.3 (so $n = N$), where the tree structures are inherent in the definition of the transformation matrix A. Although this class technically contains the Jordan-Wigner transform, our motivation is to obtain mappings that are more akin to the Bravyi-Kitaev transform, in order to keep parity strings short. While the Bravyi-Kitaev transform itself does this job perfectly, we will show that it cannot be reconciled with a square lattice connectivity graph: in this section, we instead develop a method to tailor mappings to preexisting connectivity graphs, and provide an algorithm with which short parity strings can be guaranteed and the operator weight bounded. Let us start by reviewing the Bravyi-Kitaev transform.

In [26], the mapping is introduced in order to reduce the weight of transformed fermionic operators to $O(\log N)$, which is an exponential improvement over the Jordan-Wigner transform. In the original paper, the (classical) encoding and decoding are defined by a partially ordering the mode indices according to some rules defined by their representation as binary numbers. Later works then developed the notion of flip, update and parity sets and provided a method to construct the binary matrices A^{-1} and A in $\log N$ steps [27, 37]. Instead of being one-dimensional, the partial order can be regarded placing all mode indices onto nodes inside a tree structure, which is the reason the mapping is sometimes referred to as binary-tree transform (even though the tree is not a binary tree). As pointed out in [29], the flip and update operators of every mode j, $(\bigotimes_{k\in F(j)} Z_k)$ and $(\bigotimes_{l\in U(j)} X_l)$, have a geometric interpretation on that tree (as will be illustrated shortly), so we would naturally like to match it with the qubit-connectivity graph. While an embedding is possible for small such trees, increasing N will make the tree outgrow the square lattice rather quickly. In fact, the binary rule implies that the node with index 2^j has exactly j children, and all nodes with indices below 2^j have fewer than j children. This means that trees with $N > 16$ modes, cannot be embedded in the square lattice where every site has 4 nearest neighbors. The tree for $N = 16$ can be found in Figure 3.11(a) and its embedding in the square lattice is presented in panel (b). This particular tree is however not the end of the story. In [29], it was argued that the Bravyi-Kitaev transform can be optimized to produce more local strings, in particular when considering Hamiltonians of locally-interacting fermions. For that purpose, the 'binary' trees are replaced with segmented Fenwick-tree structures. These structures are explicitly allowed to contain multiple trees, and the number of trees is even a parameter of the mapping. This number can range from 1 to N (the number of modes), where at N the mapping is identical to the Jordan-Wigner transform and at 1 it corresponds the Bravyi-Kitaev transform (in case N is an integer power of two). However, we can go even further and define mappings based on an arbitrary number of arbitrary trees. In particular, we can define tree structures that can be embedded on arbitrary qubit connectivity graphs, like our square lattice, and the associated mappings still yield small parity operators $(\bigotimes_{m \in P(j)} Z_m)$. Let us consider one specific connectivity graph.

We need to pick a forest (a set of trees) which in total has a number of N nodes. As each node will correspond to one qubit, the trees need to be connected to each other, and so we connect their respective roots. It is sufficient here for each root to be connected to two others, such that they are linked like a chain with their order foreshadowing some canonical ordering. We now choose a set of trees, such that the graph created by connecting them can be embedded in the actual qubit-connectivity graph. Let us now turn to the description of the mapping itself. For that purpose, we firstly need to assign an index to every node, a process for which we later will provide an algorithm, but for now let us assume we have done so in a prudent way. For the definition of the transform, it is sufficient to give a definition of all update and flip sets, as by corresponding sets $F(j)$ and $U(j)$ the matrices A and A^{-1} can be inferred column- and row-wise. For the flip set of index j, $F(j)$, we consider the node with index j and all its children in the tree it is on, i.e. all the nodes directly

connected to j on edges that lead away from the root. The update set $U(j)$ includes the node j and all its ancestors, i.e. all nodes on the direct line to the root (of the tree it is on), where the root is also included. A visual representation of these operators can be found in Figure 3.11(c), where the direction with respect to the root is indicated by arrows. Their embedded version can be found in panel (d) of the figure. Note that this means that by the encoding of this mappings, qubit j stores the parity information of mode j and all other modes whose index is beneath j in the tree.
For anticommutation relations like $[c_i, c_j^\dagger]_+ = \delta_{ij}$, it is important that

$$\left(\bigotimes_{k\in F(i)} Z_k\right)\left(\bigotimes_{l\in U(j)} X_l\right) = (-1)^{\delta_{ij}}\left(\bigotimes_{l\in U(j)} X_l\right)\left(\bigotimes_{k\in F(i)} Z_k\right), \quad (3.41)$$

which we now want to verify by the definitions of the flip and update sets. If j is any descendant of i, then the two operators overlap on two qubits, which means they commute. If it is not an ancestor, then the only case where the operators have overlap is when $i = j$, where they exactly overlap on that very qubit and anticommute.
We so far have suppressed the discussion of the parity operators, that will now lead into an algorithm for the index assigning and a bound for the operator weight. Let us assume that our forest consisted of τ trees, each of which has at most Λ levels and every node at most Γ children. We know that the operator weight of update and flip operators scales as $O(\Lambda + 1)$ and $O(\Gamma + 1)$, the structure of the parity set however now depends on the index assigned to the nodes. By a binary rule, the Bravyi-Kitaev transform manages to only involve $O(\log N)$ qubits in the parity operators, and we can devise a labeling that mirrors its principle. The parity operator of j is only the product of flip operators of $i < j$. On the other hand, multiplying the flip operator of a parent node k with all flip operators of its descendants will cancel all Z-operators but Z_k. Thus, in order for the parity operator of j to have low weight, as many nodes with labels $i < j$ as possible need to be descendants of j. Subsequently, the mapping with the smallest parity sets is characterized by a tree where every node has only one child, i.e. a vertical line. This mapping, that we recall as parity transform from [27], has however the problem of $O(N)$-weight update operators, and is thus of the same quality as the Jordan-Wigner transform. Indeed, one being characterized by a vertical line, the

other by a horizontal line (connected one-node trees), makes both mappings effectively one-dimensional. In order to minimize the weight of update and parity operators altogether, we need to reconcile the cancellation strategy with the tree structure. The idea is to involve only qubits in $P(j)$, that are children of the nodes in $U(j)$. Of course, this is not quite possible. If an entire tree only contains nodes $i < j$, then $P(j)$ will always contain the root of this tree. According to the formula (2.12), transforming $c_j^{(\dagger)}$ thus results in strings of weight $O(\tau + \Lambda\Gamma)$. Not only this, but the strings produced will also be continuous for transforms of single operators. Unfortunately, for pairs of operators like $c_i^\dagger c_j$, the strings are discontinuous on the first qubit that is both, an ancestor of i and j – a situation we cannot remedy.
The question is now how to assign the labels to the nodes such that this mapping is implemented, or in other words: given an unlabeled forest with connected roots, how can we obtain a mapping that outputs strings of weight $O(\tau + \Lambda\Gamma)$? For that purpose, we put labels 1 to N (in order) on the nodes according to the little program below.

Line 1 Consider the first tree in line.

Line 2 Choose a leaf and put a label on it.

Line 3 Check whether there are unlabeled siblings. If it does, choose such a sibling for the consideration in the following step. If not, proceed to Line 5.

Line 4 Check whether the current node is a leaf, and if it is, label it, otherwise put a label on a leaf chosen from the sub-tree of which the current node is the root. Continue from Line 3 with the last-labeled node.

Line 5 Check whether the last node considered has a parent. If there is a parent, put a label on it and continue from Line 3 with it. In case there is none, the previous node was a root, and we label it and proceed with the next line.

Line 6 If the root is the top of the last tree, the program ends, but if it is not, the next tree in line is considered and the program continues from Line 2.

By the end of the program, all nodes are labeled in a way such that the resulting mapping outputs strings of weight $O(\tau + \Lambda\Gamma)$. Note that there might be variations on how this process can turn out, since in several lines an element of choice is involved. We can now consider customized trees and root-connected forests. For instance, we can consider a perfect binary tree (a real one this time), which yields a $O(\log N)$ scaling as well. Although with such a tree, every node is only required to have three nearest-neighbors, the embedding of an arbitrarily-sized tree into a square lattice is still not possible. This is due to the children that run into each other as we expand the tree-embedding on the lattice. We hope however that for future work the tools provided in this section will help to tailor tree-based transforms directly to specific device layouts.

3.9.3 Technical details

This section is dedicated to the quantum codes in the foundation of every locality-preserving fermion-to-qubit mapping referenced in this chapter. In particular, we provide details on the features that distinguish our mappings from prior works: rather than trying to mimic the locality of the simulated system, we have focused on the quantum device and encoded fermions into local terms on its connectivity. However, since the fermionic Hamiltonian is local on a different graph, we have introduced Manhattan-distance strings to tackle this mismatch. To comply with those ideas, we have chosen the quantum codes to be planar on the square lattice, with locality and Manhattan-distance properties reflected in their logical operators. Catching up on a number of technical details omitted earlier in the text, we commence this section by showing the latter for the logical operators of the square lattice AQM. Specifically, we have claimed that each (relevant) logical operator only had small support on the auxiliary qubits – a statement we we will substantiate in Section 3.9.3.1 by decomposing fermion operators into Majoranas. Note that those Majorana operators should not be regarded as physical particles, but rather as a useful description of the model at hand. With this new tool, we then motivate the Manhattan-distance property in Section 3.9.3.2. Afterwards, we turn to BKSF and VCT in Sections 3.9.3.4 and 3.9.3.3. We review the literature implementations of those mappings and then adapt them such that the codes are planar on the square lattice layout. What is more, we show that our implementation results in logical operators with the Manhattan-distance property. We also add some points about the proper code space

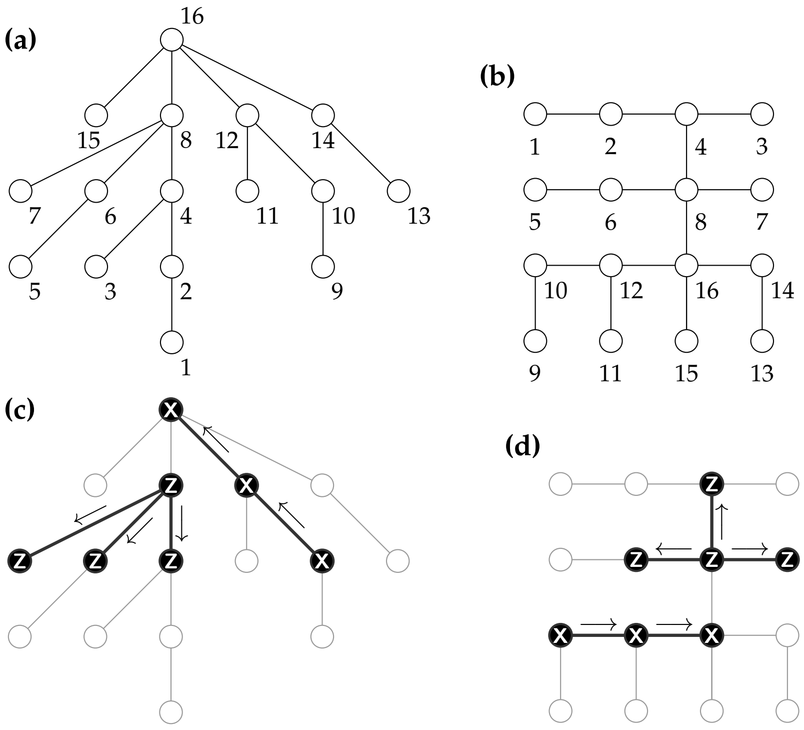


Figure 3.11. **(a)** Tree of the Bravyi-Kitaev transform for 16 qubits. Qubits are labeled from 1 to 16 according to the underlying binary tree rule. **(b)** Embedding the tree of 16 qubits into a (4×4) square lattice. **(c) & (d)** Pauli strings $(\bigotimes_{i\in U(10)} X_i)$ and $(\bigotimes_{i\in F(8)} Z_i)$ on the tree and the square lattice, where the arrows indicate the rules that determine the update set $U(10)$, and the flip set $F(8)$ respectively: $F(i)$ would involve node i and all its children, whereas $U(j)$ would involve involves node j and all its ancestors including the root.

and the logical basis, which is relevant for the logical state preparation and transformation of the Hamiltonian. For the BKSF, we describe how to constrain the simulated parity sector, and discuss the subspace of the auxiliary system for the VCT. Lastly, we consider both mappings applied to the Hubbard model, showing some of the strings referenced in Table 3.4.

3.9.3.1 Auxiliary Qubit support of the square lattice AQM

Majorana particles are fermions as their many-body wave-functions are anti-symmetric under permutation. Majorana operators $m_j^{(\dagger)}$ thus satisfy anticommutation relations like (1.7), but they are also their own antiparticles, making the operators Hermitian: $m_j^\dagger = m_j$. In general, these operators describe the relations

$$[m_i,\, m_j]_+ \;=\; 2\delta_{ij} \quad \text{and} \quad m_i m_i = 1\,. \tag{3.42}$$

For each fermionic mode, we need two Majoranas, such that the fermionic operators $c_j^{(\dagger)}$ are described by two Majorana species m_j and $\overline{m}_j$, where $\overline{m}_j$ obey the same relations (3.42), and are indistinguishable to m_j, which means $m_i\,\overline{m}_j \;=\; -\overline{m}_j\,m_i$. We define

$$c_j^\dagger = \frac{1}{2}\left(m_j - i\,\overline{m}_j\right) \quad \text{and} \quad c_j = \frac{1}{2}\left(m_j + i\,\overline{m}_j\right)\,. \tag{3.43}$$

Thus we can represent the operators m_j, $\overline{m}_j$ with the Jordan-Wigner transform as

$$m_j \mathrel{\hat{=}} \left(\bigotimes_{k=1}^{j-1} Z_k\right) \otimes X_j \quad \text{and} \quad \overline{m}_j \mathrel{\hat{=}} \left(\bigotimes_{k=1}^{j-1} Z_k\right) \otimes Y_j\,. \tag{3.44}$$

Keeping the canonical order in mind, we turn mode and qubit indices once again into coordinates just like in Section 3.5. With the index j being found at coordinate $\boldsymbol{R} = (R_1,\, R_2)^\top$, the two Pauli strings (3.44) will be denoted by $\mathcal{M}_{\text{dat}}^{b,\boldsymbol{R}}$, where $b \in \mathbb{Z}_2$ with $\mathcal{M}_{\text{dat}}^{0,\boldsymbol{R}} \mathrel{\hat{=}} m_{\boldsymbol{R}}$ and $\mathcal{M}_{\text{dat}}^{1,\boldsymbol{R}} \mathrel{\hat{=}} \overline{m}_{\boldsymbol{R}}$. It is important to note that H_{dat} is comprised of strings $\mathcal{M}_{\text{dat}}^{b,\boldsymbol{R}}$ in the same way the fermionic Hamiltonian is constructed from products of $m_{\boldsymbol{R}}$ and $\overline{m}_{\boldsymbol{R}}$. As a Hamiltonian of the form (1.8) only features products of at most four Majorana operators, the same can be said about Jordan-Wigner transformed Hamiltonians and the $\mathcal{M}$-strings. Therefore we only have to put a bound on the weight gained in the adjustment process $h_{\text{dat}} \mapsto h_{\text{dat}} \otimes \kappa_{\text{aux}}^h$ for any $h_{\text{dat}} = \mathcal{M}_{\text{dat}}^{b,\boldsymbol{R}}$. Since the adjustment is a linear process, the total weight that any term gains is four-fold that of a single $\mathcal{M}$-string. Indeed, for a coordinate $\boldsymbol{R}$ in the bulk of the qubit array, we find

$$\mathcal{M}_{\text{dat}}^{b,\boldsymbol{R}} \;\mapsto\; \mathcal{M}_{\text{dat}}^{b,\boldsymbol{R}} \otimes Z_{\boldsymbol{R}\pm\frac{1}{2}\boldsymbol{e_2}}\,, \tag{3.45}$$

where $\boldsymbol{e_2} = (0,\, 1)^\top$ is the Cartesian unit vector in vertical direction, and we recall that auxiliary are placed at half integer positions of the vertical coordinate. The point is that the adjustments only include information of one of the two auxiliaries adjacent to the data qubit at $\boldsymbol{R}$. As claimed before, this means that for hopping terms, i.e. $c_i^\dagger c_j$, the adjustments are local at the end points of the resulting strings. Note that in the case where $\boldsymbol{R}$ is at the boundary of the qubit array, there is only an adjustment such as in (3.45) if the corresponding auxiliary qubit exists.
Let us now briefly illustrate the statement (3.45). We first express the stabilizers (3.9) in terms of $\mathcal{M}_{\text{dat}}^{b,\boldsymbol{R}}$. For a stabilizer $(p_{\text{dat}}^{\boldsymbol{R}+\frac{1}{2}\boldsymbol{e_2}} \otimes X_{\boldsymbol{R}+\frac{1}{2}\boldsymbol{e_2}})$, we find

$$p_{\text{dat}}^{\boldsymbol{R}+\frac{1}{2}\boldsymbol{e_2}} = \begin{cases} i\mathcal{M}_{\text{dat}}^{0,\boldsymbol{R}} \cdot \mathcal{M}_{\text{dat}}^{0,\boldsymbol{R}+\frac{1}{2}\boldsymbol{e_2}} & \text{if } R_2 \text{ is odd} \\ -i\mathcal{M}_{\text{dat}}^{1,\boldsymbol{R}} \cdot \mathcal{M}_{\text{dat}}^{1,\boldsymbol{R}+\frac{1}{2}\boldsymbol{e_2}} & \text{if } R_2 \text{ is even}\,. \end{cases} \tag{3.46}$$

From $\mathcal{M}_{\text{dat}}^{b,\boldsymbol{R}}\,\mathcal{M}_{\text{dat}}^{a,\boldsymbol{S}} = (-1)^{1+\delta_{\boldsymbol{S}\boldsymbol{R}}\,\delta_{ab}}\mathcal{M}_{\text{dat}}^{a,\boldsymbol{S}}\,\mathcal{M}_{\text{dat}}^{b,\boldsymbol{R}}$ it is apparent that a physical operator $\mathcal{M}_{\text{dat}}^{b,\boldsymbol{R}}$ can only anticommute with a p-strings local to $\boldsymbol{R}$ or $\boldsymbol{R} - \frac{1}{2}\boldsymbol{e_2}$ (so they both exist). Since the species index has to match as well, only one of the two stabilizers will anticommute and make an adjustment (3.45) necessary. Note that it hinges on both b and the vertical component of $\boldsymbol{R}$ whether that adjustment is $Z_{\boldsymbol{R}+\frac{1}{2}\boldsymbol{e_2}}$ or $Z_{\boldsymbol{R}-\frac{1}{2}\boldsymbol{e_2}}$ in particular.

3.9.3.2 Manhattan-distance property

Verstraete-Cirac transform, Superfast simulation and the square lattice AQM - all three mappings inherently posses the Manhattan-distance property, which means that when we use them to transform hopping interaction of two fermionic modes, the weight of the (shortest) resulting Pauli string can be bounded with the Manhattan distance of the modes on the fermionic lattice. Here we will show that all mappings work in a similar fashion that enables us to use this property and elucidate why it is necessary to make use of it in a limited qubit layout. Let us recall the definition of the Majorana fermions from Section 3.9.3.1.

Majorana-pair operators like im_jm_k are used in the original proposals of VCT and BKSF, and their structure is also an element in the AQM. This is because these operators can be transformed into single Pauli strings that describe the interaction of two fermionic modes j and k, making them a useful tool for modeling it. As already established, they also have quite convenient (anti-) commutation relations. All mappings introduce extra qubits to encode operators corresponding to Majorana pairs $(m_j\, m_k) \hat{\propto} \mathcal{O}_{jk}$. In one or the other way, all mappings use these operators to prevent hopping terms, as they occur in fermionic Hamiltonians, to become nonlocal Pauli strings in the qubit Hamiltonian. When nonlocal connections of modes i with k, and k with j, as well as i with j appear in a fermionic Hamiltonian, one might think of encoding three operators $\mathcal{O}_{ik}$, $\mathcal{O}_{kj}$ and $\mathcal{O}_{ij}$. However, all mappings exhibit repercussions for adding qubits to encode these operators, such as a weight increase in the substrings κ^h_{aux} in case of the AQM. There is also the issue that we need to connect all the modes in a way that would mimic the connectivity graph of the fermionic Hamiltonian - a Hamiltonian that is generally more complicated than a lattice model. In order to be modest with the amount of qubits to be added and to be able to deal with the limited connectivity of the setup, we reconsider encoding operators $\mathcal{O}_{ij}$ of all possible combinations ij by adding qubits. Instead, under the cost of a slightly higher operator weight, we can obtain some nonlocal $\mathcal{O}_{ij}$ by multiplying operators that are already encoded: $\mathcal{O}_{ij} \propto \mathcal{O}_{ik}\mathcal{O}_{kj}$.

This is possible since for Majorana pairs we find $(m_i\, m_j) = (m_i\, m_k) \cdot (m_k\, m_j)$. We report only a 'slightly' higher weight as $\mathcal{O}_{ik}$ and $\mathcal{O}_{kj}$ have been introduced to localize their respective links in the first place. With the same argument we can take a walk over an arbitrary sequence of indices $k_1, k_2, \ldots, k_l$, where k_s and k_{s+1} are connected by an operator $\mathcal{O}_{k_s k_{s+1}}$, just to obtain the operator that links the first and the last mode k_1 and k_l

$$\mathcal{O}_{k_1 k_l} \propto \prod_{s=1}^{l} \mathcal{O}_{k_s k_{s+1}} . \tag{3.47}$$

This is the foundation for the Manhattan-distance property of all three mappings.

3.9.3.3 Verstraete-Cirac transform

Review Here we will review the Verstraete-Cirac transform starting with the original proposal [25], that, like the AQMs, can be regarded as manipulation of the Jordan-Wigner transform in which nonlocal strings are canceled with stabilizers. There, the auxiliary degrees of freedom that produce these stabilizers are added on the side of the model, where we find them in the form of Majorana modes. However, in the investigation of this mapping we found the consideration of the mapping as a quantum code more practical for a rigorous derivation of the stabilizers and outputs. This is why after a short motivation in the original language, we will describe the general concept of this mapping as a quantum code quite similar to the concept of the Auxiliary Qubit codes, which allows for the description of customized mappings such as a mapping with an odd number of rows or a qubit-economic version.

The idea of [25] is to extend the fermionic systems by doubling the number of modes, where the modes added are denoted by primed numbers from $1'$ to N'. For all indices k, k' does not denote another variable but is the primed version of the value of k. For the Jordan-Wigner transform, we need to impose the canonical order of $2N$ sites, and so we stagger primed and unprimed indices:
$1,\ 1',\ 2,\ 2',\ \dots\ N,\ N'$. Adding those primed sites, we practically increase the length of Pauli strings, since all hopping terms on the original system hop over primed sites, even turning horizontal nearest-neighbor hoppings into next-nearest neighbor interactions.

$$\begin{aligned}(i<j):\qquad c_i^\dagger c_j + c_j^\dagger c_i \;&\hat{=}\; \frac{1}{2}\left[\bigotimes_{k=i+1}^{j-1} Z_k\right](X_i\otimes X_j + Y_i\otimes Y_j)\\ \mapsto\;\; \frac{1}{2}\left[\bigotimes_{k=i+1}^{j-1}(Z_k\otimes Z_{k'})\right]&(X_i\otimes Z_{i'}\otimes X_j + Y_i\otimes Z_{i'}\otimes Y_j)\ . \qquad (3.48)\end{aligned}$$

The hopping terms are thus made sensitive to the primed subsystem, and the original system is recovered if all primed modes are empty. In their original work, Verstraete and Cirac define a fermionic quantum code, that constrains the primed subsystem completely by means of majoranic stabilizers $(i\,m_{j'}\,\overline{m}_{k'})$ for certain pairs of modes j' and k'. These are translated to the qubit side by Jordan-Wigner transform $(i\,m_{j'}\,\overline{m}_{k'})\,\hat{=}\,\mathcal{P}_{jk}$. While in the original proposal, the majoranic stabilizers $(im_{j'}\overline{m}_{k'})$ are fixed as gap terms in the model Hamiltonian, it is suggested in [55] to prepare the entangled state by making syndrome measurements with the transformed stabilizers $\mathcal{P}_{jk}$.

Stabilizers like $(im_{j'}\overline{m}_{k'})$ are useful to cancel nonlocal connections between j and k. Let us here assume that such a stabilizer is present, then the hopping between those modes can be modified by multiplication of the corresponding fermionic terms in the model Hamiltonians:

$$\begin{aligned}\left(c_j^\dagger c_k + c_k^\dagger c_j\right) i\, m_{j'}\, \overline{m}_{k'} \mathrel{\hat{=}} & -\frac{1}{2}\, X_j \otimes X_{j'} \otimes Y_k \otimes Y_{k'} \\ & + \frac{1}{2}\, Y_j \otimes X_{j'} \otimes X_k \otimes Y_{k'} \,. \end{aligned} \tag{3.49}$$

As one can see, the re-sized parity string has been canceled. Although all operators involved satisfy the correct (anti-)commutation relations, it is not possible to attribute the correct sign to all stabilizers and Hamiltonian terms without considering the code space. To do so, we now derive the quantum code version of the VCT, starting by the constructing the logical basis, that has to determine the adjustments to the Jordan-Wigner-transformed Hamiltonian terms. Although it was recently pointed out in [55], that keeping the stabilizers majoranic is unnecessary, we will stick to the original concept and merely add the freedom to 'flip' the stabilizer by introducing a sign

$$\mathcal{P}^{b_s}_{\alpha_s \beta_s} \mathrel{\hat{=}} (-1)^{b_s}\, i\, m_{\alpha'_s} \overline{m}_{\beta'_s} \,, \tag{3.50}$$

where $\boldsymbol{\beta}$, $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_r) \in [N]^{\otimes r}$ and $\boldsymbol{b} = (b_1, b_2, \dots, b_r) \in \mathbb{Z}_2^{\otimes r}$ are sequences that parameterize the mapping. A 'flipped' stabilizer would practically be implemented by requiring that syndrome measurements have the outcomes (-1), so a stabilizer $\mathcal{P}^{b_s}_{\alpha_s \beta_s}$ constrains the code space to $\langle \mathcal{P}^{0}_{\alpha_s \beta_s} \rangle = (-1)^{b_s}$. Instead of the primed and unprimed subspace to host indistinguishable fermions and being interleaved in the canonical order, we separate those modes (qubits) in an attempt to regard the primed subspace as the auxiliary register. The aux-register is not even required to have size N, instead a smaller number of auxiliary qubits can be chosen, $r \leq N$. Although separated into different registers, each auxiliary qubit is still affiliated with a data qubit, or rather their corresponding modes are. Our intention is to keep the previous notation and let the auxiliary register contain the primed labels. For that purpose, we introduce the set W as a r-sized subset of the mode numbers, $W \subseteq [N]$, such that the auxiliary register is comprised of qubits labeled $(\bigcup_{k \in W} k')$. In this way every data qubit $k \in W$ has an auxiliary qubit k' associated with it. Let us now characterize a general version of this mapping. We consider the $(\ell_1 \times \ell_2)$ block of data qubits and for every $s \in [r]$ connect the qubits α_s and β_s in a directed graph. For every qubit k that is a vertex of this graph, we add an auxiliary qubit k' somewhere, and the number k

becomes a member of W. Generalizing (3.50), the stabilizers of the qubit system are

$$\begin{aligned}\mathcal{P}^{b_s}_{\alpha_s\beta_s} &= (-1)^{b_s}\left(\bigotimes_{j=\alpha_s+1}^{\beta_s} Z_j\right)\otimes Y_{\alpha'_s}\otimes\left(\bigotimes_{\substack{k\in W\\ \alpha_s<k<\beta_s}} Z_{k'}\right)\otimes Y_{\beta'_s} \qquad \text{if} \quad \alpha_s<\beta_s\\ &= (-1)^{b_s}\left(\bigotimes_{j=\beta_s+1}^{\alpha_s} Z_j\right)\otimes X_{\beta'_s}\otimes\left(\bigotimes_{\substack{k\in W\\ \beta_s<k<\alpha_s}} Z_{k'}\right)\otimes X_{\alpha'_s} \qquad \text{if} \quad \alpha_s>\beta_s\,.\end{aligned} \tag{3.51}$$

Note that for the quantum code, that we intent to construct with the set $\{\mathcal{P}^{b_s}_{\alpha_s\beta_s}\}_{\boldsymbol{\alpha},\boldsymbol{\beta}}$ as stabilizer generators, certain conditions on $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are to be met. While these conditions are intrinsically fulfilled for the mappings in [25], we want to briefly spell them out for the sake of generality. The following conditions must be met by the directed graph on which $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are defined: (i) the graph must be composed of closed loops on the $(\ell_1 \times \ell_2)$-grid. (ii) The loops do not overlap in their vertices. (iii) The loops are uniformly directed, which means that within one loop no two edges point towards the same vertex.

Statement (i) is just a consequence of the fact that we need to constrain the auxiliary system completely. As the stabilizers (3.51) are associated with edges, we need to consider closed loops, otherwise degrees of freedom remain undetermined. We also need to make sure that all stabilizes commute and so, considering (3.50), we find that every vertex can host one incident and one outbound edge. This, together with statement (i), explains statements (ii) and (iii). An example of such a mapping, for which all three statements hold, is depicted in Figure 3.12(a), where we consider two loops in counter-clockwise directions. While in (a), we eliminate some arbitrary nonlocal connections, Figure 3.12(b) exhibits the original proposal, where the stabilizer implement the vertical connections. Of course we need to involve a few horizontal connections in order to comply with statement (i). As loops cannot be closed in it, the original proposal deals with an odd number ℓ_1 in ignoring the last column. Alternatively, we suggest that one could just create loops between vertically adjacent modes in that last column, like it is done in the right-most loop in panel (a). It is of course only possible to stabilize roughly half of all vertical connections in this way, i.e. all even or all odd pairs. Assuming an underlying S-pattern of the canonical ordering, nothing else would be required, since half of the links are local anyways. The original proposal yields a decent mapping already, as we can shorten vertical hoppings along the last column by multiplications stabilizers of the second-to-last column. In fact, the idea that not every column needs to have their own auxiliary qubits is the foundation for qubit-conserving versions of the VCT, as is shown in Figure 3.12(c). Note that in order to comply with the three statements, the periodicity $\mathcal{I}$ has to be chosen such that $(\ell_1 - 1)/\mathcal{I}$ is an odd number, the size of the auxiliary register subsequently becomes $r = \ell_2 + (\ell_1 - 1)\ell_2/\mathcal{I}$. Note that a loop of one vertex is counterproductive, resulting in a stabilizer $\mathcal{P}^b_{jj} = (-1)^{1+b} Z_{j'}$. This only fixes the parity of the auxiliary qubit, which renders it redundant since it is not entangled with the rest of the system. Not just that, it blocks the mode from being part in another loop.

Let us now take a look to the basis of the extended system. As before, the N original modes describing the fermionic Fock space shall make up the data qubit register and the primed auxiliary qubits be in the register $\text{aux} = (\bigcup_{k \in W} k')$. An ansatz for a logical basis stabilized by all $\{\mathcal{P}^{b_s}_{\alpha_s \beta_s}\}$ is

$$|\boldsymbol{\omega}\rangle_{\text{dat}} \;\mapsto\; \propto \left(\sum_{\boldsymbol{\mu} \in \mathbb{Z}_2^{\otimes r}} \prod_{s=1}^{r} \left[\mathcal{P}^{b_s}_{\alpha_s \beta_s} \right]^{\mu_s} \right) |\boldsymbol{\omega}\rangle_{\text{dat}} \otimes |\boldsymbol{\chi}\rangle_{\text{aux}} \,, \tag{3.52}$$

where $|\boldsymbol{\chi}\rangle_{\text{aux}} = (\bigotimes_{k \in W} |\chi_k\rangle_{k'})$ is a product state on the auxiliary register that can be chosen inside a certain range of parity constraints, which we now want to explain.
These parity constraints are related to a certain freedom in the characterization of the mapping. We have not determined $\boldsymbol{b}$ yet, as up to now the only restrictions we had were on the choice of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. In order to understand the role of $\boldsymbol{b}$, let us for a moment assume that the graph spanned by $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ is only one loop, which means that $\beta_s = \alpha_{s+1}$ and $\beta_r = \alpha_1$. No matter the number of loops, the sum in the basis (3.52) will always contain the product of all stabilizers around a closed loop, here it is $(\prod_{s=1}^{r} \mathcal{P}^{b_s}_{\alpha_s \beta_s})$, met by the summand for which $\boldsymbol{\mu} = (1)^{\otimes r}$. In fact, half of the terms in the sum will differ from the other half only by these operators: (having omitted the normalization factor for that reason) it is alright for some stabilizers to be linearly dependent, as long as they stabilize $|\boldsymbol{\omega}\rangle_{\text{dat}} \otimes |\boldsymbol{\chi}\rangle_{\text{aux}}$. Since we are stabilizing a loop, we find by (3.50),

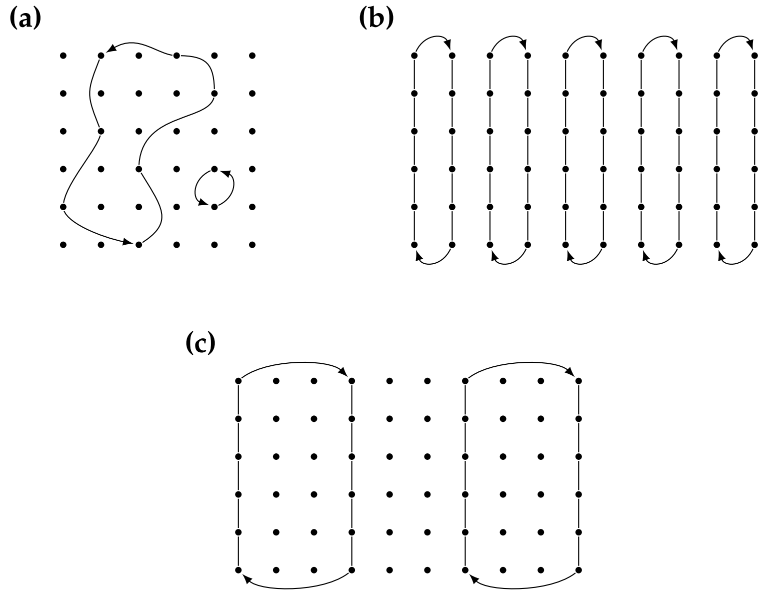


Figure 3.12. Verstraete-Cirac transform. **(a)** An arbitrary mapping showcasing the constraints on the VCT code space. The black dots correspond to data qubits. Directed loops of operators $\mathcal{P}^{b}_{jk}$ are drawn into this grid, where the direction of one loop is indicated by arrows. With 9 vertices involved, we entangle 9 auxiliary qubits to that system. **(b)** Graph of the original proposal [25]. **(c)** One possibility for a qubit-economic version of the VCT.

that

$$\prod_{s=1}^{r} \mathcal{P}^{b_s}_{\alpha_s \beta_s} = (-1)^{1+\sum_{k=1}^{r} b_k} \bigotimes_{j\in W} Z_{j'} \, . \tag{3.53}$$

Since (3.53) acts only on $|\chi\rangle_{\mathrm{aux}}$, it becomes apparent that $\boldsymbol{b}$ determines the parity of all auxiliary qubits associated with the loops in the mapping. According to the choice of $\boldsymbol{b}$, we now need to pick a state $|\chi\rangle_{\mathrm{aux}}$ that meets all parity constraints (3.53). Since we in general have more than one loop in our mapping, we need to fix the parity on several distinct subsets of $|\chi\rangle_{\mathrm{aux}}$. For instance if we pick the parity of every loop to be even, we can choose $|\chi\rangle_{\mathrm{aux}} = |0^r\rangle_{\mathrm{aux}}$.

We lastly show that Z-strings on the primed qubits come naturally as adjustments to Hamiltonian terms h_{dat}, together with minus signs from the loop parity constraints. The data-qubit substring of the stabilizers (3.51) is purely a Z-string, so we do not need to adjust a string $h_{\text{dat}} \in \{\mathbb{I}, Z\}^{\otimes N}$. This means that it is sufficient to consider the changes to be made to a string $(\bigotimes_{j=1}^{k-1} Z_j) \otimes X_k$, in order to describe all fermionic operators $c_k^{(\dagger)}$. This string anticommutes with all stabilizers, that have data qubit substrings $(\bigotimes_{j=s}^{t} Z_j)$, where $s \leq k$. These stabilizers, $\mathcal{P}^b_{(s-1)t}$ or $\mathcal{P}^b_{t(s-1)}$, act on the aux-register as

$$(-1)^b \, Y_{(s-1)'} \otimes \left(\bigotimes_{\substack{j\in W \\ t<j<(s-1)}} Z_{j'} \right) \otimes Y_{t'}$$

$$\text{or} \qquad (-1)^b \, X_{(s-1)'} \otimes \left(\bigotimes_{\substack{j\in W \\ t<j<(s-1)}} Z_{j'} \right) \otimes X_{t'} \,, \tag{3.54}$$

which means they change the parity of the subsystem that is spanned by all auxiliary qubits with the labels j', where $j \leq (k-1)$ and $j \in W$. The total parity of all auxiliary qubits is however constant i.e. it does not change with the multiplication of either stabilizer. The total parity is predetermined by $|\chi\rangle_{\text{aux}}$ and the action of a Majorana-pair operator conserves it.

If we now multiply $(\bigotimes_{j=1}^{k-1} Z_j) \otimes X_k$ to a basis element (3.52), we can determine whether it anticommutes with an even or odd number of stabilizers as we move it to the right until it reaches $|\omega\rangle_{\text{dat}} \otimes |\chi\rangle_{\text{aux}}$: it anticommutes with an odd number of stabilizers if the parity of the subsystem, spanned by all auxiliary qubits with labels at most as large as $(k-1)'$, is changed. We therefore extract the parity of said subsystem by the operator $(\bigotimes_{j\in W<k} Z_{j'})$ and add a minus sign in case $(\bigotimes_{j\in W<k} Z_{j'}) \, |\chi\rangle_{\text{aux}} = (-1) \, |\chi\rangle_{\text{aux}}$. We hence find

$$\left(\bigotimes_{i=1}^{k-1} Z_i \right) \otimes X_k \;\mapsto\; \pm \left(\bigotimes_{j=1}^{k-1} Z_j \right) \otimes X_k \otimes \left(\bigotimes_{j\in W<k} Z_{j'} \right) \tag{3.55}$$

where the sign is determined by $|\chi\rangle$. When we consider the planar code of the original proposal, we find that string has become

$$\pm \left[\bigotimes_{j=1}^{k-1} (Z_j \otimes Z_{j'}) \right] \otimes X_k \tag{3.56}$$

which is the expected string with perhaps a minus sign, depending on whether we have flipped any stabilizers. Note however that the loop parity constraints have to be fulfilled somewhere, either by minus signs in the logical operators or by flipping stabilizers.

Adaption to the layout & Manhattan-distance property We here adapt the Verstraete-Cirac transform to the square lattice connectivity, such that it has the Manhattan-distance property. In doing so, we will not stray too far from the original proposal, that is built upon the connectivity graph in Figure 3.12(b). The layout is roughly motivated by an S-pattern of the qubits ordered $1\ 1'\ 2\ 2'\ \ldots N\ N'$. For reasons that become clear later, we need the rows to be connected vertically by the auxiliary qubits, which leads us to shift every second row in order to align the primed qubits. The vertical connections are also placed along the windings of the S-pattern, resulting in a graph that can be studied in Figure 3.13(a). For the initialization of a state, stabilizers that are horizontally adjacent are multiplied pairwise. We fully constrain the auxiliary systems by those localized stabilizers, plus the stabilizers that are local already: the ones along the windings and the horizontal connections in the first and ℓ_2-th row. The stabilizer tiling to the layout of Figure 3.13(a) is presented in panel (b) of the same figure. As already remarked in [25], the analogy of the stabilizer tilings of this code and the rotated surface code [17] comes to mind easily. The tiles of the VCT are identical to the surface code on the primed qubits, but the stabilizers contain some additional Z-strings on the data qubits. Also, not all of the stabilizers might have the same sign according to b in the definition $\mathcal{P}^b_{jk}$. Curiously, only the first qubit of the data register is not entangled with the auxiliary system in any way.

Using the interpretations of the stabilizers (3.50), we can define $\mathcal{O}_{jk} \propto (-1)^b\ \mathcal{P}^b_{jk} Z_{k'}$ and obtain arbitrary long-range vertical connections over the sequence of vertically aligned stabilizers $\mathcal{P}^{b_s}_{k_s k_{s+1}}$, where $\boldsymbol{k} \in [N]^{\otimes l}$ and $\boldsymbol{b} \in \mathbb{Z}_2^{\otimes l}$, via (3.47):

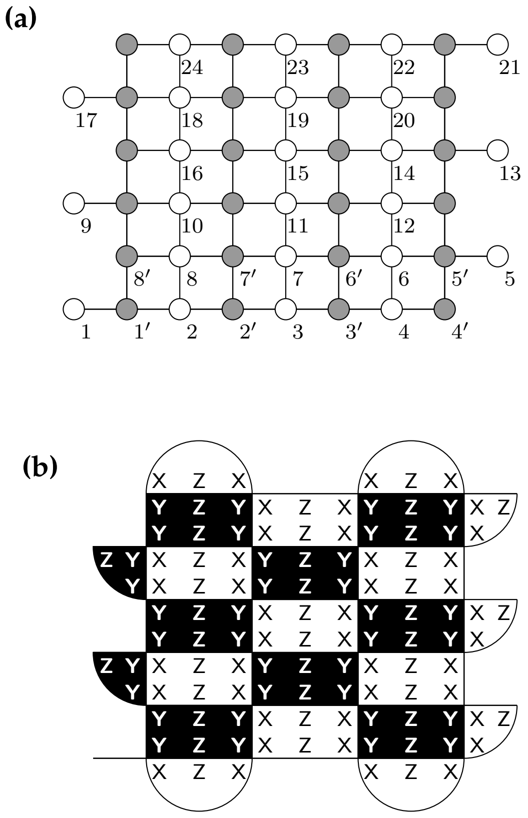


Figure 3.13. VCT as a planar code. **(a)** Connectivity graph, in which we alternate data (white) and auxiliary qubits (gray), but shift every second row such that the auxiliary qubits align vertically. The labeling of the qubits follows an S-pattern. **(b)** Stabilizers of the VCT for a graph as in Figure 3.12(b), the original proposal. We here give the connectivity graph a two-coloring of the stabilizer plaquettes, where the Pauli operators, that make up each stabilizer, are denoted by letters inside the plaquettes close to where their corresponding qubits are. Note that we have not indicated the signs that each stabilizer possibly has attached to it.

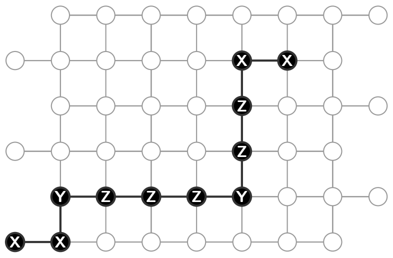


Figure 3.14. Simulating the term $(im_{20}\,\overline{m}_1)$ via the VCT, where we have arbitrarily deformed the string by the multiplication of stabilizers.

$$\prod_{t=1}^{l-1} \mathcal{P}^{b_t}_{k_t k_{t+1}} = \mathcal{P}^{a}_{k_1 k_l} \bigotimes_{u=2}^{l-1} Z_{k'_u}\,, \tag{3.57}$$

where $a = (\sum_{s=1}^{l} b_s)$. Equation (3.57) means that the multiplication of these vertical stabilizers yields a nonlocal connection $\mathcal{P}^{a}_{k_1 k_l}$, which (is not a stabilizer and) is missing the operators $Z_{k'_u}$ for $1 < u < l$. The absence of these Z-operators does not cancel them in Pauli strings originating from fermionic terms like $c_i^\dagger c_j$, where $i \leq k_1 < k_l \leq j$. These operators subsequently serve as connection between the qubits labeled k'_1 and k'_l, as the qubits are vertically aligned by our layout. With this building block we can multiply various stabilizers and so connect the qubits i and j via different paths but with the same number of gates. In Figure 3.14, we present an example of such a term.

3.9.3.4 Superfast Simulation

Review We here review the original proposal of the Bravyi-Kitaev Superfast simulation, [26], which includes the transform of the operators and the structure of the stabilizers.

In contrast to the other mappings, the Superfast simulation is not defined to transform fermionic operators, but pairs of Majoranas. Thus the BKSF only allows us to conveniently consider Hamiltonians that conserve the fermionic parity i.e. are comprised of operator pairs $c_j c_k$, $c_j^\dagger c_k^\dagger$ and $c_j^\dagger c_k$. By the relations (3.43), these Hamiltonians can then be expressed using only the operators

$$\mathcal{A}_{jk} \hat{=} -i\, m_j\, m_k\,, \tag{3.58}$$

$$\mathcal{B}_k \hat{=} -i\, m_k\, \overline{m}_k\,, \tag{3.59}$$

where $\mathcal{A}_{jk}$ and $\mathcal{B}_k$ are some Pauli strings. Using these operators, fermionic Hamiltonians can be transformed via

$$c_j c_k \hat{=} \frac{i}{4}\left(\mathcal{A}_{jk} - \mathcal{A}_{jk}\mathcal{B}_k + \mathcal{B}_j\mathcal{A}_{jk} - \mathcal{B}_j\mathcal{A}_{jk}\mathcal{B}_k\right)\,, \tag{3.60}$$

$$c_j^\dagger c_k^\dagger \hat{=} \frac{i}{4}\left(\mathcal{A}_{jk} + \mathcal{A}_{jk}\mathcal{B}_k - \mathcal{B}_j\mathcal{A}_{jk} - \mathcal{B}_j\mathcal{A}_{jk}\mathcal{B}_k\right)\,, \tag{3.61}$$

$$c_j^\dagger c_k \hat{=} \frac{i}{4}\left(\mathcal{A}_{jk} - \mathcal{A}_{jk}\mathcal{B}_k - \mathcal{B}_j\mathcal{A}_{jk} + \mathcal{B}_j\mathcal{A}_{jk}\mathcal{B}_k\right)\,. \tag{3.62}$$

The BKSF is furthermore not based on the Jordan-Wigner transform, so $\mathcal{A}_{jk}$ and $\mathcal{B}_k$ are not going to be obtained by transforming the right-hand side of (3.58) and (3.59) under (3.44). Instead, the $\mathcal{A}$- and $\mathcal{B}$-operators will be defined on a unique qubit layout, that we now introduce.
The Hamiltonian that we want to simulate describes a certain graph of pairwise interactions between modes, for example there is an edge between vertices j, k when it contains at least one of the term (3.60)-(3.62). The qubit connectivity graph of the Superfast simulation is then the line graph of this Hamiltonian graph. Here the operators $\mathcal{A}_{jk}$ are associated with edges in the Hamiltonian graph, i.e. interactions of the Hamiltonian, and the operators $\mathcal{B}_k$ are associated with vertices, i.e. fermionic modes. Let E be the set of undirected edges of the Hamiltonian graph, and ε_{jk} a number associated to the index pair jk, that yields zero if $jk \notin E$. By means of ε_{jk} a direction on the graph is fixed by imposing that if $jk \in E$, then $\varepsilon_{jk} = 1$, in case the edge is directed from $j \mapsto k$, and $\varepsilon_{jk} = -1$ when the direction is opposite. With that construction, we will take into account that $\mathcal{A}_{jk} = -\mathcal{A}_{kj}$, which is straightforward to see from (3.58). Also, on every vertex k, we need to impose an ordering of the edges connected to it. To that end Bravyi and Kitaev introduce the symbolic operator $\underset{k}{<}$, such that two different edges jk, $lk \in E$, $j \neq l$ on vertex k are ordered by a relation like $jk \underset{k}{<} lk$. As we place the qubits on the edges of that graph, both jk and kj shall be identifiers for the same qubit (given $\varepsilon_{jk} \neq 0$). In the original BKSF, the number of qubits equals the number of edges in the graph, so the qubit requirements do not depend on the system size,

but on the size of the Hamiltonian. The operators $\mathcal{A}_{jk}$ and $\mathcal{B}_k$ are defined by

$$\mathcal{B}_k = \bigotimes_{a:\, ak\in E} Z_{ak}\,, \tag{3.63}$$

$$\mathcal{A}_{jk} = \varepsilon_{jk} X_{jk} \left(\bigotimes_{b:\, bk \underset{k}{<} jk} Z_{bk} \right) \left(\bigotimes_{c:\, jc \underset{j}{<} jk} Z_{jc} \right) . \tag{3.64}$$

As shown in [26], these operators fulfill all algebraic relations that we would expect from representations of (3.58) and (3.59) but one. As it is now, the mapping would allow a Majorana to unphysically interact with itself via hopping terms around a closed loop. For a length-l sequence $a_1,\, a_2,\, a_3,\, \dots, a_l$, that describes a closed loop along edges, i.e. $a_j a_{j+1} \in E$ and $a_1 = a_l$, we must impose that

$$(i)^l \prod_{j=1}^{l-1} \mathcal{A}_{a_j a_{j+1}} \tag{3.65}$$

is a stabilizer of the system. As not all closed loops are linearly independent, one needs to stabilize only the smallest closed loops of the system.

Adaption to the layout & Manhattan-distance property We now adapt the Superfast simulation to the square lattice layout and give it the Manhattan-distance property. As we are interested in simulating more than square lattice Hamiltonians, we are going to depart a bit from the original concept of the qubit connectivity being related to the Hamiltonian.

Instead, we will show that we can adapt the mapping adequately by pretending that the Hamiltonian graph is a square lattice. On this lattice, modes that are actually subject to hopping interactions in the Hamiltonian, should be locally close. Such a lattice of modes is shown in Figure 3.15(a), where the direction of every edge is indicated. As the direction of every edge jk only determines the factor $\varepsilon_{jk} \in \{+1,\, -1\}$ in (3.63), it has not much influence on the transformation. We will see later that the choice of the order of the edges on every mode is way more relevant for the strings that such a mapping produces. In 3.15(a), we have already outlined the tiling of the line graph, to which we now switch. The resulting qubit connectivity graph can be seen in Figure 3.15(b), where the

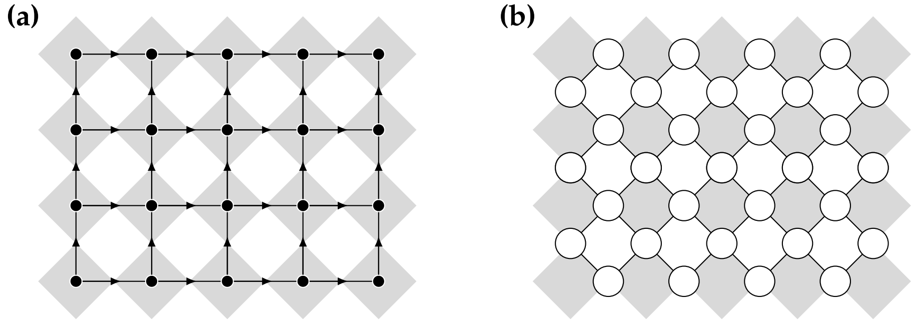


Figure 3.15. Connectivity graphs for Superfast simulation in limited connectivity. **(a)** Hamiltonian graph: all vertices correspond to fermionic modes, and in the original setting all edges would indicate the presence of hopping terms between the two modes in the Hamiltonian. We have displayed the direction of every edge in this graph. **(b)** Qubit connectivity graph: a qubit is placed on each vertex of this rotated square lattice. The underlying checkerboard pattern indicates which qubits are associated with which fermionic modes. Each dark plaquette is associated with an index k, such that the qubits on each of its corners have indices $jk \in E$.

plaquettes enclosing a fermionic mode are darkened. Starting from a general set of $\ell_1 \times \ell_2$ modes, we have now ended up with a rotated patch of the square lattice that has $2\ell_1\ell_2 - (\ell_1 + \ell_2)$ qubits on it. The number of white plaquettes, that are enclosed in the graph, describes the number of smallest possible loops, which means it is the total number of linearly independent stabilizers. We have $(\ell_1 - 1)(\ell_2 - 1)$ of those white plaquettes, which means the system has $2^{\ell_1\ell_2 - 1}$ degrees of freedom left: since we have mapped only pairs of operators (3.60)-(3.62), we are now seemingly stuck in the subspace with an even number of fermions. This situation is however not terminal: we can simulate the odd-parity subspace separately as well as the entire Fock space. Let us further illuminate this issue by considering the logical basis of the even-parity subspace first. For that purpose we pick a set $\{S^i\}_i$ of $(\ell_1 - 1)(\ell_2 - 1)$ linearly independent stabilizers from (3.65). The set fully constrains the system. Automatically, all stabilizers S^i are orthogonal in the computational basis, such that the fermionic vacuum state is encoded as

$$|\Theta\rangle \;\hat{=}\; \left[\prod_i \frac{1}{\sqrt{2}} \left(\mathbb{I} + S^i\right)\right] |0^n\rangle \;. \tag{3.66}$$

We can then apply operators $\mathcal{A}_{jk}$ and $\mathcal{B}_j$ in order to prepare other states with an even particle number. While the $\mathcal{A}_{jk}$ are different for every ordering, the operators $\mathcal{B}_k$, are independent of it: an operator $\mathcal{B}_k$ is the string of Z-operators around the shaded plaquette associated with mode k. If this plaquette is in the interior of the lattice in Figure 3.15(b), the string has weight four, three if it is on the boundary edge, and two if in a corner. The one feature that the operators $\mathcal{A}_{jk}$ have in common for every ordering, is that they include an X-operator on the qubit (jk). Apart from the administration of some minus signs, the $\mathcal{A}_{jk}$ has generally the effect to flip qubit (jk) in the all-zero state $|0^n\rangle$ of (3.66). Comparing the encoded operators (3.58) and (3.59) to the toy picture of the $\mathcal{A}$ and $\mathcal{B}$ operators we have just suggested, we find that a qubit configuration $|\boldsymbol{\xi}\rangle = (\bigotimes_{jk \in E} |\xi_{jk}\rangle_{jk})$, with all $\xi_{jk} \in \mathbb{Z}_2$, has the following correspondence to a fermionic quantum state:

$$\left[\prod_i \frac{1}{\sqrt{2}} \left(\mathbb{I} + S^i\right)\right] |\boldsymbol{\xi}\rangle \;\; \hat{\propto} \;\; \left[\prod_{j=1}^{N} \left(c_j^\dagger\right)^{\sum_{i:\,(ij)\in E} \xi_{ij} \mod 2}\right] |\Theta\rangle \, . \tag{3.67}$$

Note that (as denoted by $\hat{\propto}$) we have not kept track of any minus signs in (3.67). The relation is however sufficient to show that a fermionic mode k is occupied, if an odd number of qubits around the plaquette k are in $|1\rangle$. The product of the stabilizers $\prod_i \frac{1}{\sqrt{2}} \left(\mathbb{I} + S^i\right)$ mixes all possible configurations that conserve the common parity of qubits around a shaded plaquette (as the stabilizers need to commute with $\mathcal{B}_k$, a logical operator), and so the fermionic occupations are conserved as well. In order to prepare a pure fermionic state different from the vacuum, we need to consider a qubit configuration $|\boldsymbol{\xi}\rangle$, in which we flip strings of adjacent qubits in order to create fermions on the plaquettes at their ends, see Figure 3.16.

So far, we still have not left the even-parity subspace, but we might have systems to solve that are populated by odd numbers of fermions. In [57], it is suggested to add another mode to the system that is however not coupled to any other term in the Hamiltonian. From the original concept of the BKSF it is however not clear how this mode is brought into the system, since all qubits correspond to couplings of modes in the Hamiltonian, which here do not exist. Let us suggest to couple this mode to exactly one other, without ever using the $\mathcal{A}$-operator of this link in the Hamiltonian. For state preparation we however can have strings that end at that outer plaquette, creating a mode that does not play a role and

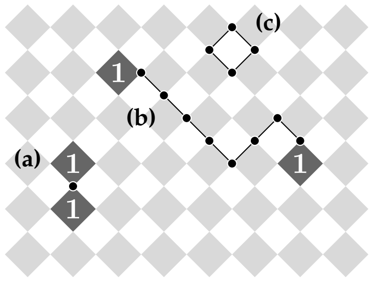


Figure 3.16. State preparation in the Superfast simulation. Black dots are flipped qubits and plaquettes with an odd number of flipped qubits are marked with **1**, as a fermion is created on the corresponding mode. **(a)** Flipping a qubit with label (jk) creates fermions on the adjacent modes j and k. **(b)** X-strings (here emphasized by linking the qubits) create nonlocal pairs of fermions, as long as we ensure to flip always an even number of qubits on each plaquette, which means winding around white plaquettes when the string has to change direction. **(c)** Flips like this result from stabilizers, and do not excite fermions, as on all dark plaquettes an even number of qubits is flipped.

so effectively increase the degrees of freedom to 2^N, modeling the entire Fock space. The cost of this increase is the overhead of one qubit. Alternatively there is a way to only map the odd-parity subspace without using additional quantum resources: the idea is to consider the plaquette k as being switched to 'filled', such that the configuration on the right-hand side of (3.66) does not correspond to the vacuum state (which is in the even-parity subspace), but to the state $c_k^\dagger \left|\Theta\right\rangle$. Flipping the qubit (jk) will lead to the fermion on k being annihilated and re-created on j, a string of flips that ends at k will in general move the fermion to the other end. We therefore make the replacement $\sum_i \xi_{ik} \mapsto (1 + \sum_i \xi_{ik})$ in the exponent of the mode-k creation operator $c_k^\dagger$ on the right-hand side of (3.67). In order to account for the switched occupation, we also need to update $\mathcal{B}_k \mapsto (-1)\mathcal{B}_k$ and add minus signs to some $\mathcal{A}$-operators.
After having established an abstract idea of BKSF on the square lattice, we will now consider different versions of this mapping as we delve into detail. As mentioned before, the stabilizers of this mapping roughly flip qubits around white plaquettes. Due to (3.65), their exact structure is determined by the operators $\mathcal{A}_{jk}$, which on the other hand depend on the ordering of edges on every vertex in the Hamiltonian graph, Figure 3.15(a). In the qubit graph, this means that with every shaded plaquette

we associate numbers with the qubits on its edges. The decision for an ordering has to be made consciously, as it influences the weight of strings simulating long-range hoppings. For now let us consider two different versions of this mapping in Table 3.7. For each version we assume that the ordering on every dark plaquette (leaving out missing vertices at the boundaries) is the same. From (3.63), we therefore just need to differentiate between vertical and horizontal version of the operators $\mathcal{A}_{jk}$, i.e. considering the directions of the edges, we need to separate the cases where (1) the plaquette k is the right neighbor of the plaquette j and (2) where the plaquette j is below k. In the Table 3.7, we sketch these operators, along with the stabilizers that follow from the multiplication of four of those operators to describe a closed loop around a white plaquette. The first version is the one already considered in [29], and second one is related to the mapping in [54].

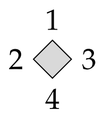
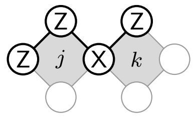
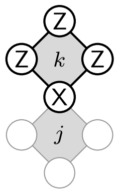
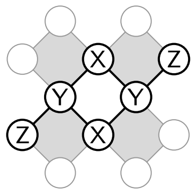
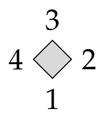
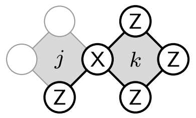
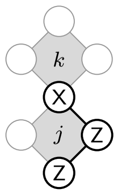
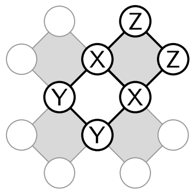
Ordering	$\mathcal{A}_{jk}$ (horizontal)	$\mathcal{A}_{jk}$ (vertical)	Stabilizer
			
			

Table 3.7. Different versions of BKSF. The ordering of the edges on each vertex is displayed as well as the operators this ordering entails: horizontal and vertical edge operators $\mathcal{A}_{jk}$ and the stabilizers (signs are omitted). The upper version is the one used in [29], while the lower one is related to the mapping in [54].

We can now describe fermion-operator-pairs via Table 3.7 with (3.60)-(3.62). The latter equations hold for operators $\mathcal{A}_{jk}$ of every link, whereas the table only provides us with operators in which j and k are adjacent plaquettes. We will now cease to pretend that the Hamiltonian is just composed of nearest-neighbor interactions, and derive nonlocal operators $\mathcal{A}_{jk}$. By (3.58) we set $\mathcal{A}_{jk} \propto \mathcal{O}_{jk}$ and using (3.47) we find

$$\mathcal{A}_{k_1 k_l} = (i)^{l-1} \prod_{s=1}^{l-1} \mathcal{A}_{k_s k_{s+1}} \tag{3.68}$$

for any sequence $k_1,\, k_2,\, \ldots,\, k_l$, where for all $s \in [l-1]$: $k_s k_{s+1} \in E$. This means we can multiply several of the nearest-neighbor $\mathcal{A}$-operators from Table 3.7. The choice of the ordering turns out to be crucial, as for various orderings, the resulting mapping is not a good one according to the criteria of Section 3.3. The first mapping in Table 3.7 for instance does not produce a continuous Pauli string (3.68) when making a chain of several horizontal $\mathcal{A}_{jk}$. For a vertical chain, we have a maximal operator weight. The second mapping on the other hand is better behaved: horizontal and vertical $\mathcal{A}$-operators are connected and their weight is minimal. In Figure 3.17, we present an example of the simulation of the Pauli string $(-i\mathcal{B}_{k_1}\mathcal{A}_{k_1 k_l})$, where $\mathcal{A}_{k_1 k_l}$ is nonlocal as in (3.68), with $l = 13$. The string here extends on a zig zag line along the edges of the plaquettes involved, $\{k_s\}_s$ connecting the plaquettes k_1 and k_{13}. The weight of this string can perhaps be optimized in cutting more corners like at plaquette k_5. In any case, we have adapted the BKSF as a two-dimensional fermion-to-qubit mapping on the square lattice.

3.9.3.5 Fermi-Hubbard model

In this section we test the proposed square lattice implementations of the Superfast simulation and the Verstraete-Cirac transform on the Fermi-Hubbard model.

For both mappings, we have to decide where to place spin-up and -down modes of the same spatial site. On the one hand should the qubits representing these modes be locally close, perhaps even horizontally or vertically adjacent, but on the other hand they will increase the weight of the strings simulating hopping terms, as they are 'in the way'. For the BKSF, it is almost inconsequential whether the spin pairs are vertically or horizontally stacked, so we decide for the latter. For the VCT, the situation is

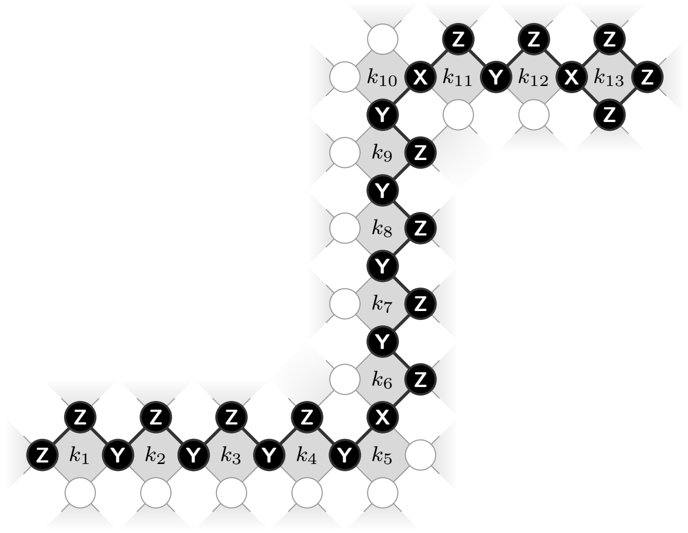


Figure 3.17. Superfast simulation of a hopping operator in the between modes k_1 and k_{13}, coupling the respective shaded plaquettes in a string of length scaling with their Manhattan distance, where the path taken is defined by the locally connected chain of modes k_2 to k_{12}. The string simulated is $(-i\mathcal{B}_{k_1}\mathcal{A}_{k_1 k_{13}})$, which in Jordan-Wigner transform would be $h_{\text{dat}} = (X_{k_1} \otimes Z_{k_1+1} \otimes \cdots \otimes Z_{k_{13}-1} \otimes X_{k_{13}})$. The plaquettes $(k_1, \ldots, k_{13})$ are labeled on this lattice.

different as it produces shorter hopping strings in the vertical direction, which leads us to make the spin pairs vertical neighbors on the grid. In order to do that, we need to compensate for the shift that has emerged aligning the primed qubits: in Figure 3.13(a), qubit 4 is for instance below qubit 6, not qubit 5. Without this shift, there would be additional costs for horizontal or vertical hoppings, but with the shift, additional costs emerge for the Hubbard terms. As a fix, we simulate the model with ℓ_2 additional modes, that remain empty. The qubits corresponding to those modes are the ones at the horizontal perimeter of the qubit lattice, i.e. the qubits labeled 1, 5, 9, 13, 17 and 21 in Figure 3.13(a). Those data qubits, fixed to $|0\rangle$, can as well be removed, but their primed counterparts must remain and be part of the code. The spin-partners can now be placed vertically adjacent on the grid. The Hubbard model with $L \times L$ spatial sites is thus simulated with $4L^2 + 2L$ qubits in the VCT, and with $4L^2 - 3L$ qubits in the BKSF. The resulting Pauli strings can be found in Table 3.8.

3.10 Notations

[...]	The set of integers from 1 to the argument.
$(i,\, j)$	Spacial coordinates replacing qubit labels in Section 3.5. To distinguish data from auxiliary qubits, the latter are given half-integer coordinates.
$\hat{=}$	Sign signifying the equivalence between fermionic operators/states to qubit counterparts according to some fermion-to-qubit mapping.
$A,\, A^{-1}$	The two binary $(N \times N)$ matrices defining linear fermion-to-qubit mappings, see Section 2.3 in the first chapter.
AQM	Auxiliary Qubit Mapping [43].
aux	The auxiliary qubit register, which comprises the integer labels from $(N+1)$ to $(N+r)$.
BKSF	Superfast simulation of fermions on a graph, also known as Superfast Encoding, by Bravyi and Kitaev [26]
$c_j^{\dagger}\,, c_j$	Fermionic creation and annihilation operators on mode j.

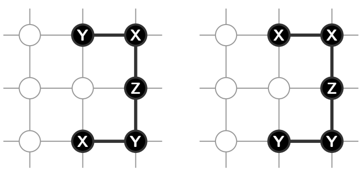
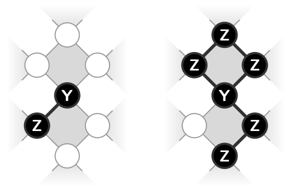
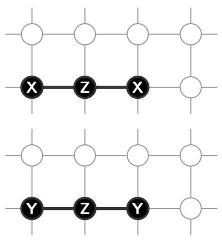
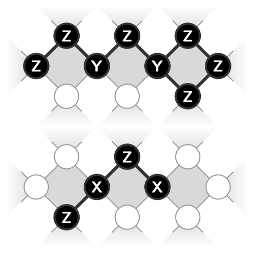
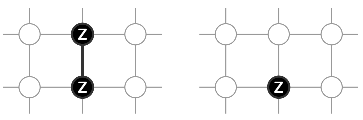
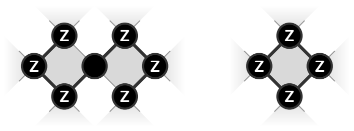
Verstraete-Cirac transform	Superfast simulation
Vertical hoppings	
	
Horizontal hoppings	
	
Hubbard interactions	
	

Table 3.8. Transforming terms of the Hubbard model according to the Verstraete-Cirac and Superfast simulation mapping. For the hoppings, we consider the real hopping terms, i.e. transforms of $(im_j\,\overline{m}_k)$ and $(im_k\,\overline{m}_j)$ for $j<k$. Note that for the Verstraete-Cirac transform, the vertical hopping terms are different for even/odd rows and columns. Here the south east qubit is in an even column and odd row. The qubit marked, but not labeled with X, Y or Z, is skipped.

$\textsc{CNot}(i \to j)$	$\|0\rangle\langle 0\|_i + \|1\rangle\langle 1\|_i \otimes X_j$. The Controlled-Not gate, where i is the control and j is the target qubit.
dat	The data register labels, which comprises all integers from 1 to N.
$F(j)$	The flip set of mode j, see (2.12).
h_{dat}, $\widetilde{h}_{\text{aux dat}}$	N-qubit Pauli strings and their logical equivalents on $N+r$ qubits. Note that $\widetilde{h}_{\text{aux dat}}$ and $(h_{\text{dat}} \otimes \kappa^h_{\text{aux}})$ are equivalent and identical by the multiplication of stabilizers.
H_{dat}, $\widetilde{H}_{\text{aux dat}}$	An N-qubit physical Hamiltonian (3.3), as for instance obtained by Jordan-Wigner transform, and its $(N + r)$-qubit logical equivalent (3.7).
H_j	$\frac{1}{\sqrt{2}}\left[\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right]$. The Hadamard gate on qubit j.
$\mathbb{I}$	The identity matrix or operation in various spaces.
$\mathcal{I}$	A parameter determining the periodicity in the sparse AQM, see Table 3.1.
κ^h_{aux}	A Pauli string on the auxiliary register. Adjustments made to the physical operator h_{dat}, see (3.6).
ℓ_1, ℓ_2	Parameters of the lattice. $\ell_1 \times \ell_2$ is the dimension of the fermionic lattice depicted in Figure 3.1(a).
L	A lattice size. $2L \times L$ is the dimension of the Fermi-Hubbard-lattice in Section 3.6.
m_j, $\overline{m}_j$	The two types of Majorana operators associated with the same mode j, see (3.42)-(3.44).
n	$N + r$. The number of qubits.
N	$\ell_1 \times \ell_2$. The number of fermionic modes.
$P(j)$	The parity set of mode j, see (2.12).
p^i_{dat}	The part of the stabilizers $(p^i_{\text{dat}} \otimes \sigma^i_{N+i})$ that is supported on the data register, see (3.5).
r	The number of auxiliary qubits.
R	A binary $(N \times N)$ matrix, in which the lower triangle is filled with ones, see see (2.13).
σ^i_{N+i}	The part of the stabilizers $(p^i_{\text{dat}} \otimes \sigma^i_{N+i})$, that is supported on the auxiliary register, see (3.5).

$U(j)$	The update set of mode j, see (2.12).
$V_{\text{aux dat}}$	The unitary initializing the code space, see (3.5).
VCT	Verstraete-Cirac transform, also known as Auxiliary fermion mapping, [25] .

3.11 Further work

Generalized Superfast Encoding

Not long after we finished [43], the Superfast simulation was extended in [74] and [75], focusing on the error mitigation properties. Both works not just manage to eliminate the blindness against certain type of Pauli errors, but even define distance three codes with which all Pauli errors of weight one can be corrected. Note that the goal of error mitigation and even correction is somewhat orthogonal to locality efforts. For quantum error correction the code distance, the minimal weight of a logical operator, is sought to be increased, whereas it is sought to be minimized for locality. However, the construction of [75], called *Generalized Superfast Encoding* (GSE) turns out to be a versatile scheme that can be used for either purpose. For the sake of completeness within this work, let us quickly sketch the idea of this mapping. In the GSE, the entirety of qubits is sectioned into partitions $n_1, n_2, \dots, n_N$, for each mode, where each partition n_j has $d_j/2$ qubits, with d_j being the number of edges at node (mode) j that the partition represents. In [75], a different encodings for the $\mathcal{A}$- and $\mathcal{B}$-operators are found relying on Majorana-like Pauli strings on every node. The strings of one node, $\{\gamma^k_{n_j}\}_{k\in[d_j]}$, only anticommute with each other, not with the γ-operators of a different node. Therefore, we can think of every node as hosting a local set of Majoranas distinctly encoded on the qubits at the node. An operator $\mathcal{A}_{jk}$ now is defined not just with the direction ε_{jk}, but also which two γ-strings on nodes j and k are associated with the edge. Any γ-string can only participate in one edge operator, and so we say that for an edge jk, we select the Majoranas s and t from nodes j and k. The logical operators are defined as

$$\mathcal{A}_{jk} \;=\; \varepsilon_{jk}\, \gamma^s_{n_j} \otimes \gamma^t_{n_k} \tag{3.69}$$

$$\mathcal{B}_k \;=\; (-i)^{d_k/2} \prod_{m=1}^{d_k} \gamma^m_{n_k}\,, \tag{3.70}$$

and the stabilizers are defined as before. In contrast to the BKSF, an interaction graph underlying the GSE is only valid if every of its vertices has even degree. That also means that there is an Euler path connecting every vertex using each edge exactly once. The Euler path is important, since the multiplication of $\mathcal{A}$-operators along yields a stabilizer proportional to $\prod_k \mathcal{B}_k$, the parity operator. It is thus the proportionality constant ± 1 that determines the total parity of the system simulated.
To simulate systems that do not comply with the even-degree condition on the interaction graph, the introduction of dummy edges and vertices is advertised in [75]. Even more conditions on the graph connectivity have to be imposed in order to achieve a code distance of three, which would allow for quantum error correction. However, for the task of this chapter, we can also try to cast this mapping into a square lattice of qubits. Considering that the interaction graph is a square lattice, the total number of qubits would be $2\ell_1\ell_2 - 4$, where it is necessary to introduce ears at the boundaries such that every node has two qubits associated with it. The only exceptions are the nodes in the lattice corners, which only have one. Ideally, one would like to define

$$\{\gamma^1_{n_j},\, \gamma^2_{n_j},\, \gamma^3_{n_j},\, \gamma^4_{n_j}\} \;=\; \{X\otimes\mathbb{I},\, Y\otimes\mathbb{I},\, Z\otimes X,\, Z\otimes Y\} \tag{3.71}$$

on every node, where the first two operators are chosen e.g. for the two $\mathcal{A}$-operators in horizontal direction. As a consequence, a Manhattan string of $\mathcal{A}_{(i,\,j)(i+x,\,j+y)}$ would roughly be of weight $x+y$. However, assigning the qubits to a square lattice, such that all $\mathcal{A}_{(i,\,j)(i+x,\,j+y)}$ are continuous, seems impossible. Instead, we can employ a different choice of (3.71) and settle for weight $x+2y$, but with continuous strings, a planar code on the square lattice with weight-6 stabilizers. This choice is characterized by the definition of

$$\{\gamma^1_{n_j},\, \gamma^2_{n_j},\, \gamma^3_{n_j},\, \gamma^4_{n_j}\} \;=\; \{X\otimes Y,\, Y\otimes Y,\, \mathbb{I}\otimes X,\, Z\otimes Y\} \tag{3.72}$$

on every node j. In conclusion, there is a Superfast mapping with Manhattan distance strings of one preferred direction like in the AQM and VCT. Its qubit requirements are similar to the VCT, and $\mathcal{B}$-strings are encoded also by a singular Z-operator. After its state preparation, classical side computations have to aid the attachment of minus signs to $\mathcal{A}$- and $\mathcal{B}$-operators, such that the targeted parity subspace is encoded. Note that while the representation (3.72) guarantees continuous strings for long-range interactions, (3.71) can be used for error mitigation purposes, as it forms a weight-2 code.