27

Logical gates with the surface code

The authors are grateful to Earl Campbell, Andrew Cross, and John Preskill for reviewing this chapter.

Rough overview (in words)

The ability to implement an arbitrary unitary operation, either exactly or approximately, is a prerequisite for performing quantum computation. It can be achieved with unitary gates that form a universal gate set [625, 801]. A commonly considered gate set contains two Clifford gates, the Hadamard gate H and the controlled X gate CX (also known as the controlled NOT gate), and one non-Clifford gate, the $T = Z^{1/4}$ gate. One can consider other non-Clifford gates, such as the Toffoli gate CCX. Note that non-Clifford gates are essential for quantum computation, as any quantum circuit comprising only Clifford gates, state preparation, and measurement in the computational basis can be simulated in polynomial time on a probabilistic classical computer [445, 4].

Since we are interested in fault-tolerant quantum computation, we would like to implement a universal set of logical gates $\overline{H}$, $\overline{CX}$, and $\overline{T}$ on information encoded in some quantum error correcting (QEC) code, such as the surface code. We can implement these gates with a planar layout of qubits and nearest-neighbor entangling gates. To be more precise, we consider a simple architecture [535] that comprises N surface code patches, each encoding a logical qubit into the surface code with code distance d, and the routing space in between; see Fig. 27.1(a). In such an architecture, the total number of data and ancilla qubits is $O(Nd^2)$.

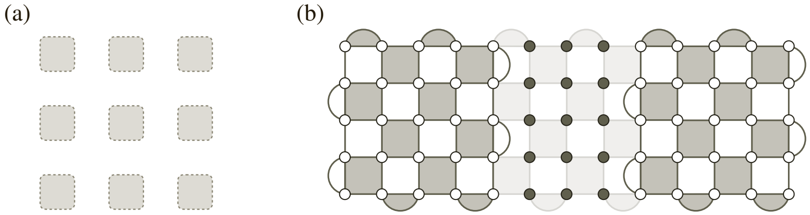
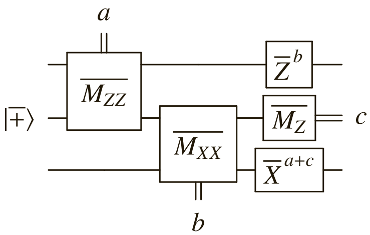


Figure 27.1 (a) A planar layout of qubits comprises surface code patches (shaded), each using the layout depicted in Fig. 26.1(a) and encoding a logical qubit, with routing space in between the patches. (b) The logical Pauli measurement $\overline{M_{XX}}$ is implemented by preparing the routing space qubits (filled dots) in the state $|0\rangle$ and repeatedly measuring parity checks (lightly shaded) in the routing space spanning between the two surface code patches. Other logical Pauli measurements, for example, $\overline{M_{ZZ}}$ and $\overline{M_{YZ}}$, require connecting different boundaries of the two patches.

Rough overview (in math)

The logical $\overline{H}$ does not pose any challenges. From a practical standpoint, it is transversal, since it can be realized by applying the Hadamard gate H to every data qubit in the surface code patch, followed by swapping of the roles of Pauli Z- and X-type parity checks in the subsequent QEC rounds. As such, the logical $\overline{H}$ takes constant time and the surface code patch is effectively rotated (which may alter how subsequent operations are implemented).

The logical $\overline{CX}$ is more challenging than the logical $\overline{H}$, since it is impossible to implement it transversally with the planar layout of qubits and nearest-neighbor entangling gates shown in Fig. 27.1(a). Instead, one can use the following quantum circuit, where the first qubit (top wire) is the control and the third qubit (bottom wire) is the target of the logical $\overline{CX}$ gate:

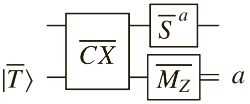


(27.1)

It is straightforward to fault-tolerantly realize preparation of the logical state $|\overline{+}\rangle$, logical Pauli measurement $\overline{M_Z}$, and logical Pauli operators $\overline{Z}$ and $\overline{X}$. In addition, the required logical Pauli measurements $\overline{M_{ZZ}}$ and $\overline{M_{XX}}$ can be implemented fault-tolerantly via "lattice surgery" techniques [535, 397, 693]; see Fig. 27.1(b) for an illustration of how to realize $\overline{M_{XX}}$. Unlike the logical

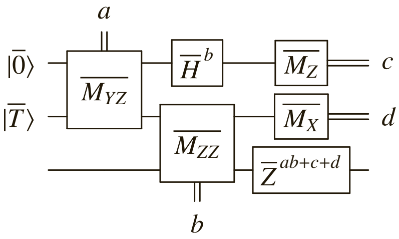
$\overline{H}$, logical Pauli measurements $\overline{M_{ZZ}}$ and $\overline{M_{XX}}$ and, subsequently, the logical $\overline{CX}$ cannot be realized in constant time; rather, due to the need to account for measurement errors, they typically incur time overhead of $O(d)$.

The logical $\overline{T}$ can be implemented using gate teleportation [448] via the following quantum circuit:



(27.2)

Here, the logical resource state $|\overline{T}\rangle$ is defined as $\left(|\overline{0}\rangle + e^{i\pi/4}|\overline{1}\rangle\right)/\sqrt{2}$, the logical gate $\overline{S}$ is defined as $\overline{Z}^{1/2}$, and the first qubit (top wire) is the control and the second qubit (bottom wire) is the target of the logical $\overline{CX}$ gate. One can fault-tolerantly implement the logical $\overline{S}$ with a planar layout of qubits [206, 423] (or even in a transversal way given access to nonlocal entangling gates [653, 788]). However, the need to apply the logical $\overline{S}$ conditioned on the measurement outcome of $\overline{M_Z}$ may slow down quantum computation, and, for this reason, it may be beneficial to use the following quantum circuit from [693, Fig. 17(b)], which is an alternative to the one in Eq. (27.2) that uses one additional logical qubit but requires only logical Pauli corrections, rather than logical Clifford corrections.



(27.3)

In either case, given the logical resource state $|\overline{T}\rangle$, the logical $\overline{T}$ typically incurs time overhead of $O(d)$. We conclude that implementing the logical $\overline{T}$ reduces to the problem of preparing the logical state $|\overline{T}\rangle$, which, in turn, can be realized via state distillation [633, 190]; see [146] for a brief overview of state distillation.

Dominant resource cost (gates/qubits)

State distillation provides a fault-tolerant method to prepare high-fidelity logical resource states, such as the logical state $|\overline{T}\rangle$. The basic idea is to convert some number of noisy resource states into fewer but, crucially, less noisy resource states. Importantly, this task can be accomplished with quantum circuits comprising only Clifford gates (together with state preparation and measurement in the computational basis) and postselection. Typically, state distillation circuits are based on some QEC code, for example, the 15-qubit Reed–Muller code.

State distillation is often described as a resource-intensive method that contributes the most to the resource overhead of fault-tolerant quantum computation with the surface code [398] (assuming many state distillation circuits working in parallel). For that reason, numerous efforts have been devoted to finding possible alternatives [189, 582, 167, 249, 146]. However, recent results indicate that state distillation may not be as costly as one may think [693, 694], especially when one allows only a few state distillation circuits to run in parallel and optimizes them for specific quantum hardware and noise that exhibits some bias [696]. In the task of estimating the ground state energy density of the Fermi–Hubbard model, state distillation of logical Toffoli resource states injected one at a time uses less than 10% of the total resources and is never a bottleneck on runtime of the quantum algorithm [250].

Often, a quantum algorithm is expressed as a quantum circuit C comprising Clifford and T gates. Thus, by using the aforementioned logical gates $\overline{H}$, $\overline{CX}$, and $\overline{T}$, we can fault-tolerantly implement the logical quantum circuit $\overline{C}$ with the surface code of code distance d and a planar layout of qubits in Fig. 27.1(a). However, from the perspective of reducing the resource overheads, it may be beneficial to consider a quantum circuit C' equivalent to the circuit C, which is obtained from C by commuting all Clifford gates to the end of C [693]. As a result, the circuit C' only comprises multiqubit Pauli $\pi/8$ rotations (which are a generalization of the T gate and can be realized via, e.g., quantum circuits analogous to the one in Eq. (27.3)). Consequently, fault-tolerant implementation of the logical circuit $\overline{C'}$ incurs qubit overhead of $O(Nd^2)$ and time overhead of $O(Md)$, where N and M are the number of qubits and T gates in C, respectively. We remark that the time overhead can be reduced at the expense of increased qubit overhead—first, by distilling more resource states and being able to use them faster, then, by implementing them in parallel [693].

Caveats

Lattice surgery is not necessary to realize fault-tolerant quantum computation with a planar layout of qubits and nearest-neighbor gates. An alternative approach (which actually preceded the development of lattice surgery) relies on the surface code with defects and braiding [862, 863, 398, 206]. However, resource overhead estimates strongly suggest that this approach is not competitive with lattice surgery [397].

The simple architecture depicted in Fig. 27.1(a) can be improved in a couple of ways to reduce the qubit overhead. First, it is possible to pack surface code patches more densely, resulting in more logical qubits for the given total number of qubits and target code distance [660, 693]. Second, one can designate certain regions, commonly referred to as magic state factories, to solely produce resource states, such as the logical state $|\overline{T}\rangle$, and optimize their design [809, 693, 694].

To simplify implementation of logical gates, one can consider other QEC codes, for example, the 3D color code [165, 648]. The gauge color code has redundant degrees of freedom, commonly referred to as gauge qubits. For different states of its gauge qubits, the gauge color code admits transversal implementation of different logical gates, which, *combined*, form a universal gate set (thus circumventing the Eastin–Knill theorem [370, 1078]). Importantly, changing the state of gauge qubits can be done fault tolerantly in constant time. However, to realize this construction one needs, for instance, a 3D layout of qubits with nearest-neighbor gates or a planar layout of qubits with a limited number of nonlocal gates, which are more challenging to engineer compared to the simple architecture in Fig. 27.1(a). To achieve code distance d with the gauge color code, one incurs qubit overhead of $O(d^3)$ (compared to qubit overhead of $O(d^2)$ for the surface code), so, similarly to single-shot QEC described in Chapter 26 on QEC with the surface code, this approach trades time overhead for qubit overhead.

Example use cases

- Lattice surgery techniques developed for the surface code can be straightforwardly adapted to, for example, the color code [659] or the surface code with a twist [1066], leading to fault-tolerant quantum computation with potentially reduced qubit overhead. Recent work [303] proposed a generalization to the setting of quantum low-density parity check codes. In addition, lattice surgery techniques can also be used for the fault-tolerant transfer of encoded information between arbitrary topological quantum codes [842].
- Now, we are ready to present a rough, order-of-magnitude estimate of the resource overheads needed to realize fault-tolerant quantum computation in the architecture based on the surface code and lattice surgery. For concreteness, we consider the circuit noise of strength $p = 0.001$, where each basic operation, including state preparation, CNOT gate, and measurement, can fail with probability p. Assume that we want to implement a quantum circuit C comprising $N = 10^3$ qubits and a certain number $M = 10^{10}$ of T gates. These resource counts are in the ballpark of estimates for various quantum algorithms in the application areas of quantum chemistry, condensed matter physics, and cryptanalysis. First, following the procedure from [693], we compile C into a new circuit C' of depth M that comprises N qubits and M multiqubit Pauli $\pi/8$-rotations implemented one at a time. Since there are

NM possible fault locations in the circuit C', the error rate for each qubit of C' should not exceed

$$\epsilon \approx 1/(NM).$$

Since each qubit of C' is realized as a logical qubit of the surface code with distance d, then its logical error rate p_{fail} can be approximated by

$$p_{\text{fail}} \approx \alpha(p/p_{\text{th}})^{d/2},$$

where we can crudely set $\alpha = 0.05$ and $p_{\text{th}} = 0.01$; see Chapter 26 on QEC with the surface code for more details. Note that these values are empirical and depend heavily on the choice of the decoder, in our case, the belief-matching algorithm [530]. Thus, in order for the logical error rate p_{fail} to reach the target error rate ϵ we need the surface code distance at least

$$d \approx \lceil 2 \log(\alpha NM)/ \log(p_{\text{th}}/p) \rceil.$$

Assuming that half of all required qubits are devoted to realizing N surface code patches (each comprising $2d^2 - 1$ data and ancilla qubits), with the other half used for resource state distillation and routing [693], we obtain that the fault-tolerant implementation of C' incurs qubit overhead of

$$n_{C'} \approx 4Nd^2$$

and time overhead of

$$t_{C'} \approx Md\tau,$$

where we crudely set $\tau = 1\mu s$ to be the time needed to implement one syndrome measurement round with the superconducting circuits architecture. Finally, our order-of-magnitude resource estimate gives 2.3×10^6 physical qubits and 67 hours of runtime. This general approach to resource estimation has been applied to a number of specific quantum algorithms in a variety of application areas; see, for example, [669, 424, 630, 147, 895]. These references often go beyond a back-of-the-envelope calculation and provide a more meticulous analysis that accounts for exact qubit layouts and the physical footprint of resource state distillation factories. They also pursue optimizations to how the circuit is implemented (e.g., exploiting space-time tradeoffs) in light of these considerations.

Further reading

- An accessible overview of fault-tolerant quantum computation based on the surface code and lattice surgery can be found in [693].

- A convenient way to describe and optimize lattice surgery operations is via the ZX calculus, which is a diagrammatic language for quantum computing [302, 335].
- A direct comparison of the resource overhead associated with preparation of the logical resource state $|\overline{T}\rangle$ using either state distillation or transversal gates (with the 3D color code) can be found in [146].
- To read about a framework for estimating resources required to realize large-scale fault-tolerant quantum computation, see [147].
- The recently introduced paradigm of algorithmic fault tolerance [1090] may significantly reduce the space-time overhead of FT quantum computation with the surface code.