# Lattice surgery translation for quantum computation



2017 New J. Phys. 19 013034

(http://iopscience.iop.org/1367-2630/19/1/013034)

IOP Publishing New J. Phys. 19 (2017) 013034 https://doi.org/10.1088/1367-2630/aa5709

New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG
IOP Institute of Physics

Published in partnership with: Deutsche Physikalische Gesellschaft and the Institute of Physics

OPEN ACCESS

RECEIVED
29 August 2016

REVISED
20 December 2016

ACCEPTED FOR PUBLICATION
5 January 2017

PUBLISHED
25 January 2017



PAPER

# Lattice surgery translation for quantum computation

**Daniel Herr**[1,2], **Franco Nori**[1,3] **and Simon J Devitt**[1,4]

[1] Quantum Condensed Matter Research Group, CEMS, RIKEN, Wako-shi 351-0198, Japan
[2] Computational Physics, ETH Zurich, 8093 Zurich, Switzerland
[3] Department of Physics, University of Michigan, Ann Arbor, MI 48109-1040, United States
[4] Superconducting Quantum Simulation Research Group, CEMS, RIKEN, Wako-shi 351-0198, Japan

**E-mail:** daniel.herr@riken.jp

**Keywords:** quantum error correction, quantum computing, surface code, lattice surgery code, quantum compiler

## Abstract

In this paper we outline a method for a compiler to translate any non fault tolerant quantum circuit to the geometric representation of the lattice surgery error-correcting code using inherent merge and split operations. Since the efficiency of state distillation procedures has not yet been investigated in the lattice surgery model, their translation is given as an example using the proposed method. The resource requirements seem comparable or better to the defect-based state distillation process, but modularity and eventual implementability allow the lattice surgery model to be an interesting alternative to braiding.

## 1. Introduction

Performing any type of quantum computation is a delicate undertaking. Quantum systems are easily perturbed due to environmental influences and/or imperfect control and thus lead to unwanted changes in the physical qubits, which will in turn lead to computational errors.

Quantum error-correction (QEC) has become an extremely well developed component of quantum information science and has shown how arbitrary quantum algorithms can be realized provided physical error rates of the hardware are below a certain threshold. While there are multiple ways to implement QEC, with numerous codes and fault-tolerant protocols available [1–3], quantum engineers need to consider hardware constraints when designing practical large-scale hardware that will not require a physically unreasonable number of physical qubits or computational time to realize an error-corrected algorithm [4–6].

The toric code [7] was the first topological quantum error-correction code discovered that had the potential for a realistic hardware implementation while also having comparatively good performance (in terms of the fault-tolerant threshold). In this code, physical spins are arranged on a two dimensional lattice with periodic boundary conditions. A set of commuting quantum operators, called stabilizers, is chosen and measured continuously. The stabilizers are defined locally over a group of four neighboring spins and the continuous measurement of the eigenvalue of these stabilizers enables the detection and correction of errors on physical spins. For a $N$-qubit toric code, there are $(N - 2)$ linearly independent stabilizers, hence there are two degrees of freedom that can be used to encode information. Thus, the toric code can be used to encode two logical qubits.

### 1.1. Toric code

The structure of the toric code (most notably the periodic boundary conditions) makes this code difficult to implement in realistic hardware. Luckily, this code can be generalized to other variants. The most common is the surface code [8–10], which is still defined on a 2D lattice of qubits, but does not require periodic boundary conditions. Additionally, the number of degrees of freedom (encoded qubits) can be greater than two, with a sufficiently large array of physical qubits, by voluntarily switching off stabilizer measurements. These are termed defects and any computation can be performed by braiding them around each other [10, 11].

**Table 1.** Overview between different topological error-correction techniques. Here, $d$ denotes the error-correcting code distance. The space–time requirements for lattice surgery are calculated at the end of this paper.

| | Toric code | Surface code | Planar code (Lattice surgery) |
|---|---|---|---|
| Encoding qubits | Loops around torus | Defects | Patch of planar code with open boundary conditions |
| Number of possible qubits | 2 | $\infty$ | 1 per patch |
| Performing computation | Memory only | Braided logic | Merge and split operations between isolated planar codes |
| Compiler to geomentric representation | Not applicable | Was devised in [29, 30] | This work |
| Optimization | Not applicable | Unsolved | Conceptually simpler |
| Overall space/time requirements for $\|Y\rangle$-state distillation | Not applicable | 140 $d^3$ [28] | 280 $d^3$ |
| Overall space/time requirements for $\|A\rangle$-state distillation | Not applicable | 1500 $d^3$ [28] | 1080 $d^3$ |
| Overall space/time requirements Bravyi–Haah $(3k + 8)$-to-$k$ state distillation for $k = 4$ | Not applicable | 4688 $d^3$ [31] | 2268 $d^3$ |

### 1.2. Surface code

The surface code itself, and its performance, has been extensively studied [12–16], and due to both its performance and comparative ease of implementation for physical hardware has recently become the code of choice for most hardware models under experimental development [17–22]. There has also been significant work related to the classical compilation and control for a quantum computer operating under this model [23–27]. Compiling fault-tolerant quantum algorithms for this model essentially consists of generating a three-dimensional geometric description of a topological circuit which has a certain space/time volume [28]. Resource optimization requires the compaction of this structure using rules that manipulate and reduce this space/time volume (and consequently the number of qubits/time required for computation) without changing the functionality of the topological circuit [25]. However, this optimization problem has so far proven to be difficult [29] and the resource cost is still unclear for fully compiled and optimized quantum circuit.

### 1.3. Planar code

A second approach to achieve universal computation with the surface code does not rely on topological braiding. This type of toric code is the planar code [32]. It allows for the encoding of a single qubit of information without periodic boundary conditions; hence the computer now becomes an array of isolated 2D patches of surface code, each representing a logical qubit. The toric code and its variants allow for a transverse application of the logical CNOT gate (a transversal CNOT gate is where corresponding physical qubits in two logical blocks are interacted via a physical CNOT gate), but a transversal logic gate eliminates the 2D nearest-neighbor geometry that is extremely desirable for hardware implementation. To mitigate this problem a technique, called *lattice surgery* [33], was developed, that re-introduces only 2D nearest-neighbor interactions to achieve a logical CNOT gate between two encoded qubits. These CNOTs are achieved by turning on (off) syndrome measurements on the boundary between adjacent qubits. Such operations are called *merges* (*splits*). A computation in this geometric structure has to perform computation using merges and splits, which can emulate gates such as CNOTs. Combining this with state-injection [34] allows for the realization of a universal set of quantum gates [33].

Circuit compilers have been developed that translate higher-level circuits into the appropriate geometric forms for braid-based topological computation [29, 30]. Preliminary steps have been made to both benchmark and optimize physical resources using this model [4], but the question remains whether, ultimately, a lattice-surgery-based approach to computation will be more resource efficient. As with braiding-based computation, we first need a generalized set of protocols to compile an appropriate fault-tolerant circuit into a form appropriate for lattice surgery, design optimization protocols for this form and ultimately compare physical resources to braiding-based approaches. In this paper we take the first step and provide a method that will convert an appropriately designed high-level circuit into a physical layout and scheduling pattern for implementation in a lattice-surgery-based topological quantum computer.

### 1.4. ICM representation

For this we start with the ICM representation [30], which is a formulation used to compile arbitrary circuits using braid-based logic, and is divided into three distinct parts: Initializations, **C**NOTs and **M**easurements. This description operates at the logical level of the computation and is designed to be compatible with all Calderbank–Shor–Steane-based error-correction codes [35, 36]. First, all qubits are initialized in one of four

distinct states. Secondly, CNOTs are applied to these qubits to perform entanglement operations; then in the last step the qubits are measured.

This formalism incorporates not only the higher-level decompositions to convert a quantum algorithm into an appropriate Clifford $+T$ gate library, compatible with fault-tolerant error-correction, but also includes ancillary protocols such as state distillation [34, 37–39], which are needed for an operational computer. Our formalism utilizes an inverse model of ICM [40], where qubit initializations are restricted to two states, $|0\rangle$ and $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, and measurements are performed in a rotated basis to achieve universality. The entangled states before their measurements in the inverted ICM representation is similar to graphs states and their creation using parity checks has been described in [41, 42]. By working in this inverted ICM representation, we can use the natural operations exhibited by the lattice surgery protocol to realize a universal set of logic operations and to layout and schedule an arbitrary quantum circuit for implementation on an actual error-corrected machine.

The inverted ICM formalism follows a similar structure to measurement-based quantum computation [43, 44]; however, our approach works at the fault-tolerant error corrected level. This formalism prepares an effective encoded graph state which is algorithmically specific given the original circuit specification. Computation then proceeds via encoded rotated-basis measurements on each qubit of this encoded graph. Our method prepares such an entangled state in a completely fault-tolerant manner during the initialization and CNOT steps and maintains the physical 2D nearest-neighbor restrictions of the underlying hardware.

The work presented in this paper is akin to the already existing compiler for the braided error-correction scheme [29]. Using this we will discuss its resource requirements on three exemplary state-distillation algorithms that we have attempted to optimize manually. As the bulk of operations in a fault-tolerant quantum computer is related to ancillary protocols, such as state distillation, numerous techniques have recently been investigated [45, 46]. To illustrate the translation, this is done explicitly, and its complexity is compared to the braiding method.

Nevertheless, the three examples of state distillation circuits in this work were implemented naively for illustrative purposes and did not take advantage of recently proposed methods of optimization [46, 47]. Additionally, there have been several papers that have further optimized the layout of encoded information using the planar code [48, 49]. These proposed techniques are compatible with lattice surgery and hence the results presented in this work can be improved upon such that the resource requirements decrease even further. However, this analysis together with further scheduling optimizations are left for future work.

### 1.5. Outline of the paper

We will now give a short outline on the structure of the paper and describe what has been done throughout the following sections. In sections 2 and 3 we review previous literature and adapt their findings such that they can be used in our translation. Afterwards, in section 4 we describe the format in which a quantum algorithm has to be represented in order to use our translation. In section 5 we outline the first part of our mapping to the fault-tolerant lattice surgery model. This description builds on the concepts presented in the previous sections. Afterwards we describe how to implement an algorithmically specific entangled state, which in many error correcting frameworks is described by the stabilizer matrix formulation. In section 6 we describe how our approach can be translated to this formalism, which allows easier comparison of the entangled states. To finish the translation to lattice surgery, we describe the procedure of measurement in section 7. This concludes the translation and three example (state distillation) circuits are investigated under the devised method in section 8. Section 9 summarizes our results.

## 2. Quantum computing

In general, error-correction schemes are agnostic of the underlying hardware. While codes are often constructed with hardware constraints in mind, any type of qubit in any type of appropriate hardware model can be used. In recent years, many advances in quantum hardware [50, 51] have been made such that lower error rates in physical qubits were archived. Thus the threshold for surface code error-correction was surpassed and the implementation of surface code corrected qubits is now, in principle, possible (there is still significant work that is needed to scale systems and maintain low error rates). This makes the current investigations for this method of error-correction an important task. It involves providing a classical framework at the hardware level [52] to develop classical algorithms to track errors and correct them if necessary [26, 27, 53] and to optimize qubit and time resources for a compiled, error-corrected algorithm [25, 28].

A universal quantum computer requires a certain, discrete, number of gates that can be used to construct any arbitrary unitary operation. While there are many universal gate sets, there are restrictions imposed by a QEC code such that some sets are preferred over others. Most importantly, each element of a universal set needs a fault-tolerant implementation on an appropriately chosen quantum code. Arguably, the most common

universal set (and the one that is used with the surface code) is the Clifford $+T$ set [54–57], which is generated by the gates [36]:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \qquad T = \begin{bmatrix}1 & 0\\ 0 & e^{i\frac{\pi}{4}}\end{bmatrix}$$
$$\text{CNOT} = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix}.$$

While not strictly required, this generating set is generally augmented with the $P = T^2$ gate (which combined with the CNOT and Hadamard gate generates the Clifford group), as there are more efficient implementations of this gate compared to using two $T$-gates in sequence [34].

It is common to write quantum algorithms in a circuit where each element is one of these operations. This is a good representation on a conceptual level, but needs to be translated to lower hardware. Such a feat is done by a quantum compiler. Among recent proposals [30, 40, 58], one proposed design stack consists of the following steps:

1. Algorithm: high-level functions like quantum Fourier transform, which consist of many applications of gates.
2. Non-fault-tolerant circuit with general gate set: this representation is akin to the description of quantum circuits in textbooks.
3. Fault-tolerant circuit with a reduced gate set: using a subset of gates and a defined structure, which can be implemented on the error-correcting code that will be used. One of these circuit classes is the ICM representation [30].
4. Geometry: this is where the fault-tolerant circuits are translated into a geometric representation for braided topological logic [29]. In this work we address this step for the lattice surgery approach using planar codes.
5. Mapping: measurement operations of syndrome qubits are switched off and on in order to perform the geometric algorithms [59].
6. Hardware: individual hardware instructions, e.g. laser pulses, that manipulate the state of single physical qubits.

Our method translates from level 3 to level 4, where a general quantum circuit is transformed to a lattice surgery error-corrected algorithm. The description of the algorithm presented in this paper is still agnostic to its underlying hardware. For defect-based surface codes these algorithms already exist and were proposed in [30]. However, the braided geometric representation has proven to be difficult to optimize [25] and has not been studied in depth.

The method proposed in this paper, albeit a complex optimization problem, appears to be more feasible to optimize than the braided version, because it is relatively close to two problems, where one is the traveling salesman problem and the other the sliding puzzle problem [60].

## 3. Computation using lattice surgery

We will now review the basic concepts of lattice surgery [33] and in section 3.4 describe a method for entanglement creation used throughout our translation. In lattice surgery each qubit is encoded in one patch of error-correcting surface code with open boundary conditions (commonly referred to as a planar code). Logical $X$- and $Z$-operators are defined as chains of physical operators that span the whole patch of code from either left to right (logical $Z$) or top to bottom (logical $X$) (figure 1). Computation can be performed by allowing interaction between different patches of code.

To interact two logical qubits, a line of data and syndrome qubits is added on the boundary between two patches of code and stabilizers between them are either turned on or off. Switching on/off these stabilizers, merging/splitting individual logical qubits, act as parity checks on the basis states of the logical qubits that partake in the operation. Using these operations natively instead of emulating the true effect of CNOT operators gives rise to more efficient computation. The four new operations are smooth/rough merges and smooth/rough splits. In the following, the effect of these operations is described, a more elaborate explanation can be found in the original paper [33].

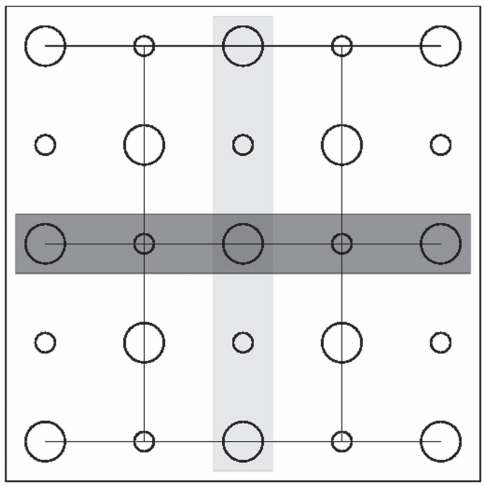

**Figure 1.** A depiction of a patch of error-corrected distance-3 surface code encoding one logical qubit. Here, the syndrome qubits are represented using small circles and data qubits are drawn in big circles. The application of a logical $X$-operator is given by performing a physical $\sigma_x$ operator on all data qubits in the light gray box. The logical $Z$-operator is performed by applying $\sigma_z$ on all data qubits inside the dark horizontal box.

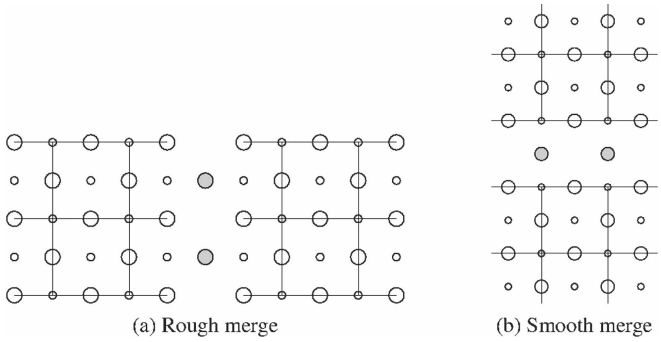


**Figure 2.** Examples of the merge/split operation in lattice surgery codes. In this case, two stabilizer measurements, represented by the two gray circles are turned on (merge) or off (split). During the merge process these intermediate gray qubits need to be prepared in either $|0\rangle$ for a rough merge or $|+\rangle$ for a smooth merge.

### 3.1. Merge operations

Figure 2 shows the difference between a smooth and a rough merge in the lattice surgery model. One can see that this difference comes from stabilizer measurements that have to be turned on as a mediator between the different patches of surface code. For a smooth merge these are $Z$-stabilizers, whereas for a rough merge $X$ stabilizers are used. As an example, we describe the rough merge process. First, the intermediate (physical) data qubits need to be prepared in the $|0\rangle$ state, then the measurements of the $X$-stabilizers along the edge between the two surfaces are turned on. These measurements will later be needed to precisely determine the state and whether a correction operator needs to be applied.

Mathematically, if the initial states are given by $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and $|\phi\rangle = \alpha'|0\rangle + \beta'|1\rangle$, the post-merge state for a rough merge evaluates to

$$\begin{aligned} |\psi\rangle \circledM_r |\phi\rangle &= \alpha|\phi\rangle + (-1)^M \beta|\overline{\phi}\rangle \\ &= \alpha'|\psi\rangle + (-1)^M \beta'|\overline{\psi}\rangle, \end{aligned}$$

where $|\overline{\phi}\rangle = \sigma_x|\phi\rangle$; and $M \in \{0, 1\}$ is the aforementioned measurement result of the intermediate stabilizers. The effect of this operation is a parity measurement on both states, which decreases the number of possible degrees of freedom by one. For the smooth merge, one has to consider a basis transformation of the pre-merge states to $|\psi\rangle = a|+\rangle + b|-\rangle$ and $|\phi\rangle = a'|+\rangle + b'|-\rangle$. The post-merge state now evaluates to

$$\begin{aligned} |\psi\rangle \circledM_s |\phi\rangle &= a|\phi\rangle + (-1)^M b|\overline{\phi}\rangle \\ &= a'|\psi\rangle + (-1)^M b'|\overline{\psi}\rangle. \end{aligned}$$

Here the state $|\overline{\phi}\rangle = \sigma_z|\phi\rangle$ is the negation in the $\pm$ basis.

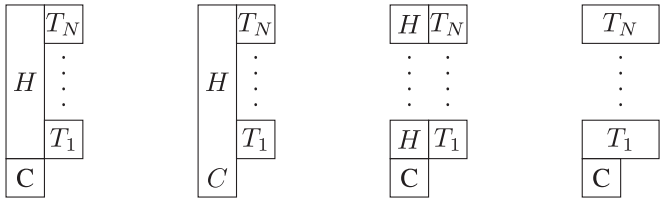


**Figure 3.** This figure illustrates the application of multi-target CNOTs in lattice surgery. The patch denoted by C corresponds to the control qubit and patches with $T_i$ denote the $i$th target. Furthermore, patches denoted by $H$ are ancillary patches. In a first step, a vertical patch composed of all ancillas is initialized to $|+\rangle$ and using a smooth merge the control qubit is merged to this patch. Afterwards, this large section is split into individual patches and in the final step each ancilla is merged to one target. This operation results in the same state as a multi-target CNOT.

### 3.2. Split operations

During a split operation a single logical qubit will be split into two. Again the boundary between the two newly created logical qubits will be used to discriminate between a smooth and a rough split. The individual qubits of this border are measured out and the two remaining surfaces are then stabilized individually.

For a smooth split we can note that the removal of the qubits on the boundary will not change the outcome of any of the joint $Z$-operators. Thus both states are in the same superposition of eigenstates of the $Z$-operator. Mathematically, the smooth split will give

$$\alpha|0\rangle + \beta|1\rangle \rightarrow_{\mathrm{s}} \alpha|00\rangle + \beta|11\rangle. \tag{1}$$

For the rough split one will get

$$a|+\rangle + b|-\rangle \rightarrow_{\mathrm{r}} a|++\rangle + b|--\rangle. \tag{2}$$

Performing a basis transformation on this last equation will give us the effect of a rough split on an arbitrary state in the $Z$-basis

$$\alpha|0\rangle + \beta|1\rangle \rightarrow_{\mathrm{r}} \alpha(|00\rangle + |11\rangle)/\sqrt{2} + \beta(|01\rangle + |10\rangle)/\sqrt{2}. \tag{3}$$

These split operations enable the creation of entanglement among the encoded qubits. One should note that after performing a split or merge one needs $d$ rounds of error-correction cycles to compensate for faulty physical measurements in the computer. However, this might be reduced by making use of the fact that the errors are likely concentrated along the boundary of the split and merge, and this is currently under investigation.

### 3.3. Multi-target CNOT

We now turn our attention to the implementation of logical operations. In the following, we introduce an implementation of fanouts or multi-target CNOT gates. These operations will form the core of our compiler. In order to properly perform a multi-target CNOT in lattice surgery, we expand the original description of [33] on how to perform CNOTs. Here, one needs an additional encoded ancilla qubit for each target qubit. In the following derivation, we consider a general control qubit given by $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. As depicted in figure 3, we prepare all encoded ancilla qubits in one single $|+\rangle$ state and perform a smooth merge between the ancilla qubit and the control qubit. After $d$ rounds of error-correction this combined qubit is split smoothly into $N + 1$ qubits, where $N$ is the number of target qubits. The resulting state is given by $\alpha|0 \cdots 0\rangle + \beta|1 \cdots 1\rangle$. Now for each of the $N$ target qubits a rough merge with one of the entangled qubits from the smooth split is performed. This results in the state:

$$|\psi\rangle = \alpha|0\rangle \otimes [(|0\rangle\langle\mathrm{M}_{\mathrm{r}}|T_1\rangle) \cdots (|0\rangle\langle\mathrm{M}_{\mathrm{r}}|T_N\rangle)] + \beta|1\rangle \otimes [(|1\rangle\langle\mathrm{M}_{\mathrm{r}}|T_1\rangle) \cdots (|1\rangle\langle\mathrm{M}_{\mathrm{r}}|T_N\rangle)].$$

The resulting state from the rough merge is given by

$$|\psi\rangle = \alpha|0\rangle \otimes |T_1\rangle \otimes \cdots \otimes |T_N\rangle + \beta|1\rangle \otimes |\overline{T_1}\rangle \otimes \cdots \otimes |\overline{T_N}\rangle$$

which is exactly the effect of a multi-target CNOT. A more resource-friendly implementation is discussed in the next section.

### 3.4. Split and merge circuit

The multi-target CNOT implementation introduced in the previous section works well but requires many ancillary patches. In this section we devise a shorthand implementation for entanglement creation by providing an example circuit in figure 4. The drawback of this implementation is, however, that it is restricted to only $|0\rangle$ and $|+\rangle$ input states. The output state of this circuit is given by

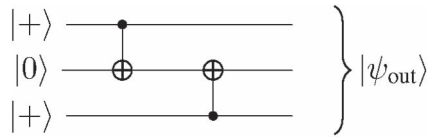


**Figure 4.** This circuit will be used as an example to see how CNOTs can be implemented in lattice surgery with less overhead. Furthermore, this can be extended to multi-target CNOTs.

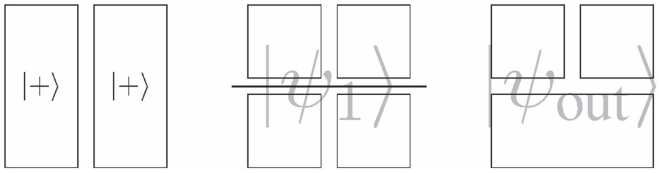


**Figure 5.** Implementation of the circuit in figure 4 using the shorthand implementation of the multi-target CNOT operation (in this case only 1 target) for qubits. Three steps are needed: first, an initialization of two patches to $|+\rangle$; these correspond to the control qubits of each CNOT. Then, two smooth splits, represented by the horizontal line in the second frame, creating two two-qubit entangled states. In the end, a rough merge between two qubits connects two patches, such that the final state of the 3 remaining patches is the same as for the circuit in figure 4.

$$|\psi_{\text{out}}\rangle = \frac{1}{2}(|000\rangle + |110\rangle + |011\rangle + |101\rangle).$$

It can be seen in the following that the application of two splits and one merge has the same effect as this circuit. A graphical representation of this is given in figure 5. Two encoded qubits are both initialized to $|+\rangle$. Since these are encoded in rectangular patches of surface code, our schematic representation of this will be boxes which can be merged and split. Both of these qubits will now be split using a smooth split. This will lead to an intermediate state given by:

$$|\psi_1\rangle = \frac{1}{2}(|00\rangle + |11\rangle) \otimes (|00\rangle + |11\rangle). \tag{4}$$

Using a rough merge between two of these patches, one will also obtain $|\psi_{\text{out}}\rangle$.

The measurement during the split operations corresponds to measuring the operator $X_L X_L$ on the two qubits. Here, the measurement outcome is completely determined, and one will always obtain $M = 1$, because the initial states are already prepared in the eigenstate $X_L = 1$. One has to note that this method only works if the qubit states are in $|+\rangle$; otherwise there will be a nonzero chance to obtain the measurement output $M = -1$. The post-measurement state of that output is not the correct entangled state and cannot be used anymore. This is the reason why the ICM representation is incompatible and has to be replaced later on by an inverted ICM representation.

Thus, one can see that any state which can be prepared with $|+\rangle$ and $|0\rangle$ and CNOTs can also be prepared using split and merge operations. Here, one does not need the logical ancilla qubit that would be required when applying the full CNOT circuit. States that can be prepared using $|0\rangle$ and $|+\rangle$ inputs and CNOT gates are subsets of states known as Calderbank–Shor–Steane stabilized states [36]. These stabilized states are characterized as having two sets of stabilizers, one consisting purely of $X$ operators and one purely of $Z$. As a further remark, this derivation can similarly be performed by using multi-target CNOTs.

## 4. ICM representation

The compilation procedure presented in this paper requires an input circuit that is given by a fault tolerant circuit with a reduced gate set (stage 3 of the proposed design stack in the introduction). To this end we will introduce the ICM model that was devised in [30]. Circuits in the ICM model perform first an initialization of all qubits then an array of CNOTs is applied and measurements are performed in the end. A sample schematic of this is shown in figure 6. Any higher-level circuit can be rewritten in this form, which includes all necessary ancilliary protocols for fault-tolerant quantum error-correction. A required gate set for the ICM representation is given by the gates: CNOT, $H$, $P$ and $T$. The main idea of the ICM model is to use quantum teleportation to implement the operators $H$, $P$ and $T$ at the cost of introducing specially prepared ancilla qubits. This allows an arbitrary circuit to be constructed with a *deterministic* number of logical qubits and array of CNOT gates. The left parts of figures 8 and 9 show two teleportation circuits. The teleportation-based circuits are probabilistic, such

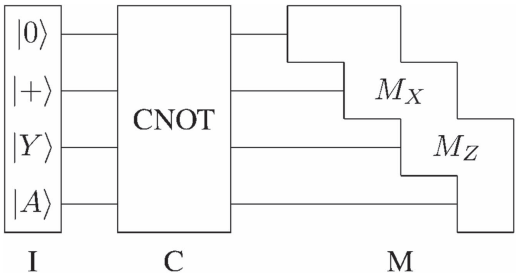


**Figure 6.** The circuit here is divided into three distinct steps. First in the initialization stage (I), the qubits will be initialized in one of the following states: $|0\rangle$, $|+\rangle$, $|Y\rangle$ or $|A\rangle$. In the second step (C), a layer of CNOTs is applied to these states. In the last step, staggered measurements (M) in the $X$- or $Z$-basis are applied, where each basis is chosen depending on the previous measurement results.

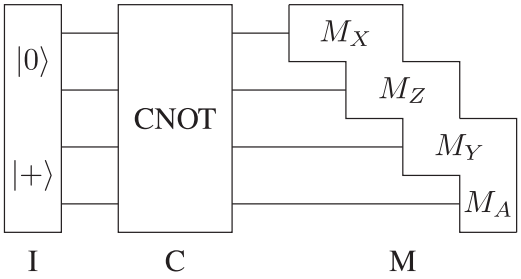


**Figure 7.** Similarly to the ICM circuit, this inverted ICM circuit has three steps: initialization (I), CNOT (C) and measurement (M). This time, the initialization step only prepares states in $|0\rangle$ and $|+\rangle$. This comes at the expense that measurements have to be performed additionally in the $Y$- and $A$-basis. This form is now compatible with the split and merge operations from lattice surgery.

that correctional gates might need to be applied after the measurement of the ancilla qubit. This would require a dynamically changing quantum circuit, depending on the measurement result of the ancilla qubits during teleportation. Yet the ICM model shows a way to mitigate this problem by introducing more ancilla qubits and performing selective target teleportation and selective source teleportation circuits [61], such that the ICM model achieves a deterministic gate array for fault-tolerant, error-corrected computation.

This form of ICM requires a simulation of CNOTs using splits and merges, which requires further ancillary regions. This is due to the requirement to initialize some logical qubits in the $|A\rangle$ or $|Y\rangle$ state for teleported $T$- and $P$-gates. We want to obtain an algorithmically specific entangled state with as few ancillas as possible. Thus, we devise a similar representation to ICM, which uses the merge and split operation natively. An example has already been given in section 3.4, but we will now formalize this such that arbitrary circuits can be implemented.

### 4.1. Inverse ICM

Using the shorthand implementation of multi-target CNOTs, a merge is only guaranteed to result in the same transformation as a full CNOT gate if the initial qubits are given in the states $|+\rangle$ or $|0\rangle$. Thus, the ICM formulation cannot be used as is and an inverted model is required, where the initialization step only prepares these two states (figure 7). A translation to this form has already been described in [40] and will be outlined here. For this discussion we will use a phase state $|\theta\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\theta}|1\rangle)$ and an arbitrary state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ to show equivalences, which then apply for $\theta = \{\pi/4, \pi/2\}$, for the $|A\rangle$ and $|Y\rangle$ states needed in a fault-tolerant implementation. In order to prove the equivalence of ICM and inverted ICM, only two cases need to be considered. These cases are given in figures 8 and 9. An inverted ICM circuit can be constructed by combining these sub-circuits, such that initialization is only in the $|0\rangle$ or $|+\rangle$ state and $P$- and $T$-gates are realized through rotated basis measurements.

The first equivalence is shown in figure 8. Depending on the measurement outcome, the output state of the second qubit is either given by $|\psi_{\text{out}}\rangle = (\alpha + e^{i\theta}\beta)|0\rangle + (e^{i\theta}\alpha + \beta)|1\rangle$ or $|\psi_{\text{out}}\rangle = (\alpha - e^{i\theta}\beta)|0\rangle + (e^{i\theta}\alpha - \beta)|1\rangle$. This holds true for both circuits shown.

The equivalence of figure 9 is not as straightforward as for the one before because the output state needs to be corrected. For the following argument we calculate the effect of the circuits only for the case of an $|A\rangle$ state. The $|Y\rangle$ state can be calculated analogously. The output states for the ICM circuit (left) are either $|\psi_{\text{out}}\rangle = \alpha|0\rangle + e^{\frac{\pi i}{4}}\beta|1\rangle$, for a $|0\rangle$ measurement, or $|\psi_{\text{out}}\rangle = \beta|0\rangle + e^{\frac{\pi i}{4}}\alpha|1\rangle$, when the measurement is $|1\rangle$. That is, if we measure $|1\rangle$, a $T^\dagger$-gate is applied instead of a $T$-gate. Thus the $P$-gate needs to be applied in order to


**Figure 8.** The circuit on the left is one part of the ICM circuit whereas the circuit on the right constitutes the corresponding inverted ICM model. These two circuits are identical for any state of the form $|\theta\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\theta}|1\rangle)$ and a general $|\psi\rangle$, which is mapped to $|\psi_{\text{out}}\rangle = R_z(\theta)|\psi\rangle$.


**Figure 9.** The ICM circuit on the left is again reformulated to the inverted circuit on the right. Depending on which state $|Y\rangle$ or $|A\rangle$ was given in the original circuit, a different correction needs to be applied. For the $|Y\rangle$ state a $X$-correction is required, while for the $|A\rangle$ state a $SX$ error occurs, which can be corrected without violating any requirements of the inverted ICM formulation. The $Y$- and $A$-basis measurements are achieved via a $P$- or $T$-gate, followed by an $X$-basis measurement.

correct for this error, as $PT^\dagger = T$. This error cannot be tracked and needs to be applied using the previously mentioned selective destination teleportation algorithms [61]. After that, the qubit is in a state of $XZ|\psi\rangle$, where the Pauli operators can be tracked classically. The $P$-gate has a similar correction, but as this is a simple $Z$-gate, $ZP^\dagger = P$, this can be classically tracked without any further correction.

The inverted ICM measurement has the advantage that this correction is not needed. For a $|0\rangle$ measurement the state is the same as for the ICM circuit. However, if the inverted ICM measurement returns a $|1\rangle$, the output state will be $|\psi_{\text{out}}\rangle = \alpha|0\rangle - e^{\frac{i\pi}{4}}\beta|1\rangle$ for the $T$-gate and $|\psi_{\text{out}}\rangle = \alpha|0\rangle - i\beta|1\rangle$ for the $P$-gate, which only requires a Pauli $Z$-correction.

These circuits can now be stacked together such that any circuit given in ICM format can be translated by removing all the rotated state initializations and replacing these by rotated measurements. However, the implementation of such measurements are not fault-tolerantly protected in the surface code and basis transformations given by $T$- and $P$-gates need to be performed. For these gates, state injection is needed. Furthermore, a $T$-basis transformation is probabilistic and requires further correctional $|Y\rangle$ states.

## 5. Implementation of I and C

Using the inverted ICM model, we can now derive an algorithm that prepares the required *logically* entangled states using purely the native split/merge operations in the lattice surgery model. The (I) and (C) parts of the circuit create a specific Calderbank–Shor–Steane stabilized state that reflects the overlying algorithm, after which rotated logical measurements are performed (in an identical way to the original description of cluster state quantum computation [43], were all Pauli measurements are made on a 2D cluster state resource, defining an algorithmically specific graph state consisting of rotated basis measurements and feedforward). Since the inverse ICM model is a universal model for fault-tolerant, error-corrected quantum computation, so is the presented model, with the proposed translation. The translated circuit relies on smooth split operations followed by rough merge operations. After this, the qubits are in an algorithmically specific encoded state, ready for measurements and feedforward to be performed.

### 5.1. Classical algorithm

The classical algorithm given as additional material can be used to translate the circuit to a representation that can be implemented by the lattice surgery model. It reduces the number of (multi-target) CNOTs to a minimum. The algorithm relies on three simple circuit-modification rules:

1. Any CNOT that targets an unentangled $|+\rangle$ has no effect, as this is the eigenstate with eigenvalue 1.
2. A CNOT whose control qubit is in $|0\rangle$ does not have any effect.
3. A CNOT with the same target and control qubits as its neighbor acts like the identity.

Furthermore, we use the commutation relations between different non-commuting CNOTs shown in figures 10 and 11.

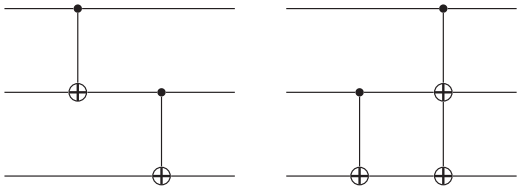

**Figure 10.** CNOT commutation relations. The circuit on the left has the same effect as the circuit on the right, such that this identity can be used to reformulate circuits.

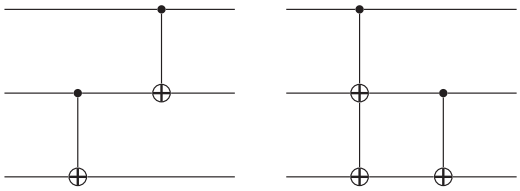

**Figure 11.** CNOT commutation relations, which can be used to reformulate circuits and achieve the form required by lattice surgery.

Using the rules above, the algorithm now proceeds as follows: in a first step, all CNOTs that target a qubit initialized to $|+\rangle$ are permuted to the beginning and using rule 1 the CNOTs are deleted. Then all CNOTs with a target qubit that initialized to the $|0\rangle$ state are also moved to the front individually and are deleted using rule 2. Due to the commutation relations given in figures 10 and 11 new CNOTs are created, but their control qubit is not located on a $|0\rangle$ state. This reduces the circuit to one where each CNOT has a control qubit initialized to the $|+\rangle$ state at the beginning. Many redundant CNOTs were introduced during the permutation actions, which need to be cleaned up. Thus in a final step all CNOTs operating on the same pair of qubits are moved together and are annihilated using rule 3. Because of rule 1 the control qubit of any CNOT is not targeted by any other CNOT such that each CNOT commutes with the others. Thus the cleanup can be performed easily.

### 5.2. Lattice surgery translation

Now any circuit can be translated to a circuit that has some number of multi-target CNOTs, whose control qubits are in the $|+\rangle$ state. A circuit of this form is easy to translate to the lattice surgery model. Its procedure requires four distinct steps. First, all the initial $|+\rangle$ states are prepared. Using smooth splits these qubits will, in a second step, be split into independent groups of entangled *logical* qubits, depending on the number of target qubits each CNOT has. In order to connect these independent blocks one will (in the fourth step) merge corresponding qubits from each independent block. This merging operation has to occur on neighboring blocks in the lattice surgery model, such that in an intermediate third step one needs to shrink or grow the size of the surface code patches or move them to another location.

An example for this translation is shown in figure 12. Here a circuit with three multi-target CNOTs is translated to the lattice surgery model with no optimization. First, a vertical patch of the surface code is initialized to $|+\rangle$ for each multi-target CNOT. The first multi-target CNOT consists of qubits 1, 6 and 7. Thus, the first patch is split into 3 patches, which can be labeled 1, 6 and 7 indicating which patch corresponds to which qubit in the circuit. It should be noted, that the ordering of these labels is arbitrary, because, so far, each patch encodes the same state. However, multiple patches corresponding to the same qubit exist (labeled by the same number), such that merges are necessary between them. This unoptimized illustration assigns each qubit an individual row, such that merges are always possible. This makes it clear that the space-cost in this representation is upper-bounded by the number of logical qubits times the number of multi-target CNOT patches. For the circuits, later on we will optimize the placement of these patches manually.

For a large quantum circuit, the placement of these logical regions within the overall computer and how they are split/merged and shifted around will dictate overall resource costs in terms of physical qubits and computational time. We illustrate three explicit examples for state distillation circuits that are extensively used in surface code computation, but the more general problem of scheduling and resource optimization using this technique is relegated to future work. A more detailed analysis, combined with recent results reducing physical resources in the lattice surgery model [46, 48, 49] is anticipated to result in better performance than topological braiding (Note: the optimization problem has not yet been solved for braiding-based logic, so an ultimate comparison will only be fair when both problems are finally solved).

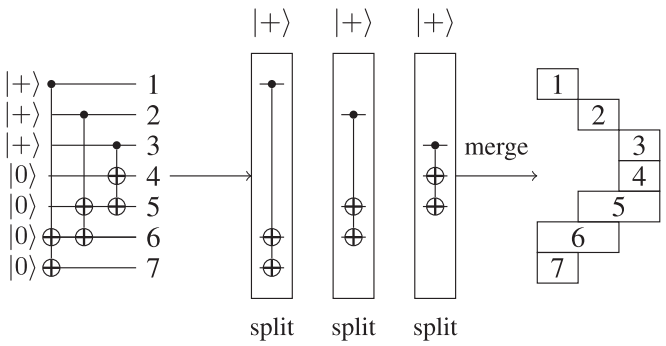


**Figure 12.** This illustrates what operations have to be implemented to perform the I and C parts of any inverted ICM circuit. First, each CNOT gets its own vertical patch of surface code, initialized to $|+\rangle$. Using smooth splits these CNOTs are split into their individual qubits. Common qubits among the different CNOTs are merged, such that the output of the circuit looks akin to the right configuration. Each number here indicates which qubit from the original circuit is encoded in which patch of surface code. In this case no effort was spent on optimization which reduces the number of steps to three: preparation, smooth splits and rough merges.

## 6. Stabilizer matrix

The inverted ICM model is designed to create an algorithmically specific Calderbank–Shor–Steane-stabilized state [35, 36] that is compatible with fault-tolerant and error-correction protocols. The split and merge operations in lattice surgery can also be re-written in terms of stabilizer transformations, which allows us to link the two in a straightforward manner.

### 6.1. Merge operation

During a merge between two encoded qubits, the number of encoded degrees of freedom decreases by one. For a rough merge the rules are given as follows

$$\begin{aligned}[X\ I] &\rightarrow [X]\\ [I\ X] &\rightarrow [X]\\ [I\ I] &\rightarrow [I].\end{aligned}$$

For the logical $Z$-operators, this action is not as easy, since it might be that stabilizers need to be merged. If there are no $Z$ stabilizers acting on the two qubits that partake in the merge operation, nothing has to be done. If only one of the qubits is affected by a $Z$ stabilizer, the post-merge rule for the merged qubit is given by

$$[Z\ I] \rightarrow [Z] \qquad [I\ Z] \rightarrow [Z].$$

For a general stabilizer matrix, many $Z$ stabilizers might act on the same qubit. Since any linear combination of these stabilizers gives an equivalent stabilizer description, one can add and subtract stabilizer rows without changing the behavior of the stabilizer. If the two qubits are acted upon using more than two $Z$ stabilizers, one can always find a representation in which only one $Z$ operator exists in each column of the merging qubits. The two stabilizer rows are merged using the following rules:

$$\begin{aligned}\begin{bmatrix}Z\\ I\end{bmatrix} &\rightarrow [Z] \qquad \begin{bmatrix}I\\ Z\end{bmatrix} \rightarrow [Z]\\ \begin{bmatrix}I\\ I\end{bmatrix} &\rightarrow [I] \qquad \begin{bmatrix}Z\\ Z\end{bmatrix} \rightarrow [I].\end{aligned}$$

### 6.2. Split operation

The physical result of a general split operation is given in equation (2) for a rough split and equation (3) for a smooth split. Due to the creation of a new encoded qubit during the process of the split operation, a new column has to be created in the stabilizer matrix. Since the created qubit is linked to the pre-split state, a new stabilizer row has to be created as well. This new stabilizer is given by $ZZ$, for a smooth split at the position of the two affected qubit positions. Furthermore, any pre-split stabilizer will transform using the following rules

$$[Z] \rightarrow [Z\ I] \qquad [X] \rightarrow [X\ X] \qquad [I] \rightarrow [I\ I]$$

for a smooth split. Exchanging $X$ with $Z$, one will get the relations for a rough merge.

### 6.3. Example algorithm

The resulting state of the example circuit in figure 4 will give the stabilizer matrix:

$$\begin{bmatrix} Z & Z & Z \\ X & X & I \\ I & X & X \end{bmatrix}.$$

Using the rules described above we will derive the stabilizer matrix using lattice surgery moves. First, we start in a state with two qubits each initialized individually into $X$-eigenstates. Then both an additional column for the additional qubit and an additional line for the stabilizer are created

$$\begin{bmatrix} X & I \\ I & X \end{bmatrix} \rightarrow \begin{bmatrix} X & I \\ I & X \end{bmatrix} \rightarrow \begin{bmatrix} X & X & I \\ Z & Z & I \\ I & I & X \end{bmatrix}.$$

After the same steps are performed on the other qubit, the middle qubits are merged.

$$\begin{bmatrix} X & X & I & I \\ Z & Z & I & I \\ I & I & X & X \\ I & I & Z & Z \end{bmatrix} \rightarrow \begin{bmatrix} Z & Z & Z \\ X & X & I \\ I & X & X \end{bmatrix}.$$

The previously mentioned procedures can now perform the (I) and (C) steps of the inverted ICM model using the inherent merge and split operations of lattice surgery. The remaining part is to illustrate how the rotated basis measurements of the inverted ICM model can be performed in lattice surgery.

## 7. Measurement step

Having prepared an entangled state using I and C of the inverted ICM representation, measurements need to be performed. But for the 2D surface-code patches used in this paper, fault-tolerant measurements in arbitrary bases are not possible. This requires the application of basis transformations which rely on the injection of magic states. The main work during the measurement step consists of the preparation and application of these basis transformations, which we direct our focus to in the following.

The method described in the following can be applied to any phase gate, but in this paper we will focus only on the $P$- and the non-Clifford $T$-gate. In surface code implementations these gates are hard to implement, as QEC does not support their immediate application. Thus the common technique used so far is state-injection, where magic states are prepared and used to perform such gates. These magic states are given by

$$|Y\rangle = \frac{1}{\sqrt{2}}(|0\rangle + \mathrm{i}|1\rangle), \tag{5}$$

$$|A\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{\mathrm{i}\frac{\pi}{4}}|1\rangle\right). \tag{6}$$

A magic state is obtained by manipulating a single physical qubit and then encoding it in the error-correcting framework. Since this process is not fault-tolerant, the error on the resultant logical state is dominated by the physical preparation of the ancilla and its encoding. To purify the state, a process called state distillation is required [34]. The main effort in the application of $P$- and $T$-gates is spent distilling magic states, such that an efficient way to perform the distillation procedure will give an efficient $P$- or $T$-gate. At the end of this paper the performance of these distillation algorithms will be compared to braiding using the translation proposed in this paper.

If a suitable (clean) ancilla qubit is available, the $P$- or $T$-gates can be applied by the circuits given in 8 and 9. In fact, any quantum computer that aims at outperforming classical hardware has to rely on non-Clifford gates, since any circuit without these can be simulated efficiently using classical hardware [62]. This makes a resource-friendly application of the $T$-gate crucial.

One should note that the implementation of a $P$-gate requires one ancilla state set to $|Y\rangle$. Since these are teleportation-based protocols, a measurement has to be performed to apply the operator. With this measurement, the magic state resource gets destroyed for both the $P$- and the $T$-gate. The destruction of $|Y\rangle$-state ancillas can be avoided using a technique [63, 64] which requires direct Hadamard operations on a planar code, which is possible using code-deformed techniques [65]. However, we do not explicitly consider this here and instead look at distillation circuit constructions for the $|Y\rangle$ state.

### 7.1. Special cases merge

Here we will explicitly calculate the post-merge state of a smooth merge between an arbitrary phase qubit $|P\rangle = |0\rangle + p \cdot |1\rangle$ and a general qubit $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$. With this we will show that both the $P$- and $T$-gates

can be efficiently implemented using a single smooth merge operation and only require as a prerequisite the magic state $|Y\rangle$ or $|A\rangle$. In the conjugate basis, the states are given by

$$|P\rangle = \left(\frac{1+p}{2}\right)|+\rangle + \left(\frac{1-p}{2}\right)|-\rangle$$
$$|\phi\rangle = \left(\frac{\alpha+\beta}{2}\right)|+\rangle + \left(\frac{\alpha-\beta}{2}\right)|-\rangle,$$

such that we can insert this into the post-merge state:

$$|A\rangle \circledM_s |\phi\rangle = \left(\frac{\alpha+\beta}{2}\right)|P\rangle + (-1)^M \left(\frac{\alpha-\beta}{2}\right)|\overline{P}\rangle.$$

If the measurement result was $M = 0$, then this evaluates to:

$$\begin{aligned}|A\rangle \circledM_s |\phi\rangle &= \left(\frac{\alpha}{2} + \frac{\beta p}{2}\right)|+\rangle + \left(\frac{\alpha}{2} - \frac{\beta p}{2}\right)|-\rangle \\ &= \alpha|0\rangle + \beta\, p|1\rangle.\end{aligned}$$

If the measurement was $M = 1$, then the state will be given by:

$$\begin{aligned}|A\rangle \circledM_s |\phi\rangle &= \left(\frac{\alpha p}{2} + \frac{\beta}{2}\right)|+\rangle + \left(-\frac{\alpha p}{2} + \frac{\beta}{2}\right)|-\rangle \\ &= \beta|0\rangle + \alpha\, p|1\rangle.\end{aligned}$$

Setting $p = i$, we obtain an implementation for the $P$-gate. If $M = 0$, the effect of this merge operation already coincides with $P|\psi\rangle$. But in this case of $M = 1$, a Pauli-$X$ and Pauli-$Z$ correction have to be applied.

Moreover, for an implementation of the $T$-gate we require $p = e^{i\frac{\pi}{4}}$. Again, if $M = 0$ no corrections are needed. Yet for $M = 1$, instead of the $T$-gate the operator $XT^\dagger$ was applied. This can be corrected by first performing a Pauli-$X$ operator and then a $P$-gate. The $P$-gate correction needs to be physically applied and cannot be tracked, and a selective source/destination teleportation algorithm has to be employed. Thus this lattice surgery implementation has the same drawbacks as the teleportation circuits when we perform the rotated-basis measurements.

One should note that the application of this merge can be thought of a basis transform which is needed in the measurement step of the inverted ICM model, where measurements in the $A$- and $Y$-basis are needed.

### 7.2. Encoding

The last part needed for universal computation is how to encode one of the magic states in a lattice surgery patch. For completeness, a summary of the required steps from the original papers [33, 66, 67] is given in the following.

First a physical qubit is prepared in the desired state. All the other qubits of the error-correcting surface code need to be initialized to the state $|0\rangle$.

Using the CNOTs that are required for the measurement of the planar code stabilizers, this state can be transformed to a superposition state involving the original data qubits and the syndrome qubits immediately above and below it. These syndrome qubits can now be swapped with the data qubits on the opposite side of the original qubit. This results in a state where a vertical line of three data qubits in the error-correcting surface code are initialized to the state

$$|\psi\rangle = \alpha|000\rangle + \beta|111\rangle. \tag{7}$$

Now the stabilizers for a distance-three error-correcting surface patch around these specially prepared qubits can be turned on, and thus an error-corrected state is obtained. The distance of these states can be increased by performing merge operations with neighboring regions. Here, the neighboring qubits need to be initialized to $|0\rangle$ again.

## 8. Examples

This concludes the description of the compilation procedure. Any circuit that has been translated to the inverse ICM representation can now be implemented in the planar code using efficient operations that the 2D lattice of surface code patches enables. After initialization, the CNOT step is entirely implemented using merge and split operations; and the last step of measurements uses state distillation and injection as described in the section 7. In the following, we provide a translation to lattice surgery for three common distillation protocols: the 7-qubit Steane code, the 15-qubit Reed–Muller code, and finally a Bravyi–Haah distillation code. For all of these codes we provide a hand-optimized placement of the surface code patches and estimate their resource requirements.

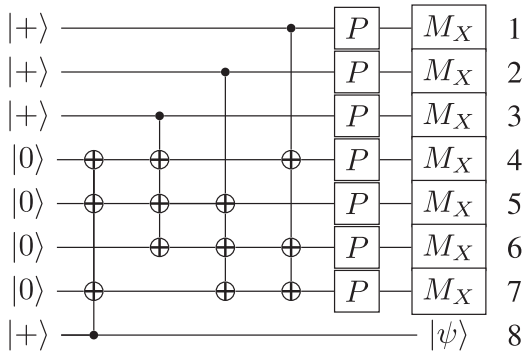


**Figure 13.** This circuit is the Steane code, to be used for the distillation of $|Y\rangle$ states. This is an iterative procedure where the error-prone $|Y\rangle$ are used during the application of the $P$-gates. The numbering used in this circuit coincides with the numbering of the algorithm given in figure 14.

### 8.1. Steane code for $|Y\rangle$-state-distillation

As a first example, we show how the state distillation algorithm for the $P$-gate can be translated to the lattice surgery model. The underlying error-correcting code for the distillation is a 7-qubit Steane code given in figure 13. One can see that this is already given in the inverse ICM format, if one replaces the $P$-gates and $X$-basis measurements by a rotated measurement.

For this algorithm, eight encoded qubit regions are needed, which are in the beginning prepared in an entagled state using the multi-target CNOT operations. This distillation circuit now applies to seven encoded qubits an error-prone $P$-gate. After that, the ancilla qubits are measured and depending on the outcome this circuit will have a distilled version as output $|\psi\rangle = |Y_{k+1}\rangle$, where $(k+1)$ denotes the output as a distilled $|Y\rangle$ state at $(k+1)$ levels of concatenation. This algorithm can be used recursively until the desired precision is reached.

At the measurement stage, we are checking the eigenvalues of the three stabilizers

$$S_1 = M_X^1 M_X^4 M_X^6 M_X^7$$
$$S_2 = M_X^2 M_X^5 M_X^6 M_X^7$$
$$S_3 = M_X^3 M_X^4 M_X^5 M_X^6.$$

Only if all of these stabilizers return the trivial syndrome, then the distillation procedure works. Otherwise, the state is discarded and another distillation run has to be performed. Furthermore, if the product of all measurements is $M_X^1 \cdots M_X^7 = 1$, then the output state is given by $|\psi\rangle = \sigma_z|Y\rangle$, whose $Z$-error needs to be tracked.

Since the multi-qubit CNOTs of the Steane code prepare the system in a Calderbank–Shor–Steane-stabilized state, one can easily translate this to split operations in the lattice surgery model. No further classical translation is required. Using the method described before, each CNOT corresponds to an instance of smooth splits, which will then be connected by rough merges. The placement of the individual qubits can be treated as an optimization problem in order to place corresponding qubits close to each other.

A method to proceed is given in figure 14. The procedure starts with the initialization of four encoded regions to the state $|+\rangle^{\otimes 4}$. Because smooth splits will be performed, these four regions have sufficient size to accommodate four encoded regions each. Afterwards, in figure 14(b), the smooth splits are performed creating all qubits that are needed in the original circuit given in figure 13. In the next step, qubit one will be moved one space down and the leftmost qubit 7 will be moved to the right. After that, all patches contributing to the same qubit are merged using a rough merge. The result is visualized in figure 14(c).

At this point, we have prepared the entangled state between eight logically encoded regions that reflect both the initialization and CNOT parts of the distillation circuit. The remaining operation is to perform each individual logical $P$-gate on qubit regions one to seven and measure them out in the $X$-basis. For each of these qubits we need to introduce ancillary $|Y\rangle$ states. Without loss of generality, we assume that this is a level-one concatenated circuit, end hence we need to state-inject seven physical $|Y\rangle$ states and encode them into encoded regions. To free up lattice space, we first shrink qubit regions seven, six, four, and five and use the resulting lattice space to inject and encode $|Y\rangle$ states which are adjacent to the encoded regions that are needed in figure 13. Finally, smooth merges are performed and the resulting qubits will be measured such that only logical qubit 8 remains, which is our output, with the rest of the code space now free to be used in subsequent circuits.

Using this procedure recursively will exponentially decrease the errors associated with the magic state. Therefore, one can obtain an arbitrary precise $Y$-state and thus the application of a arbitrarily precise $P$-gate is

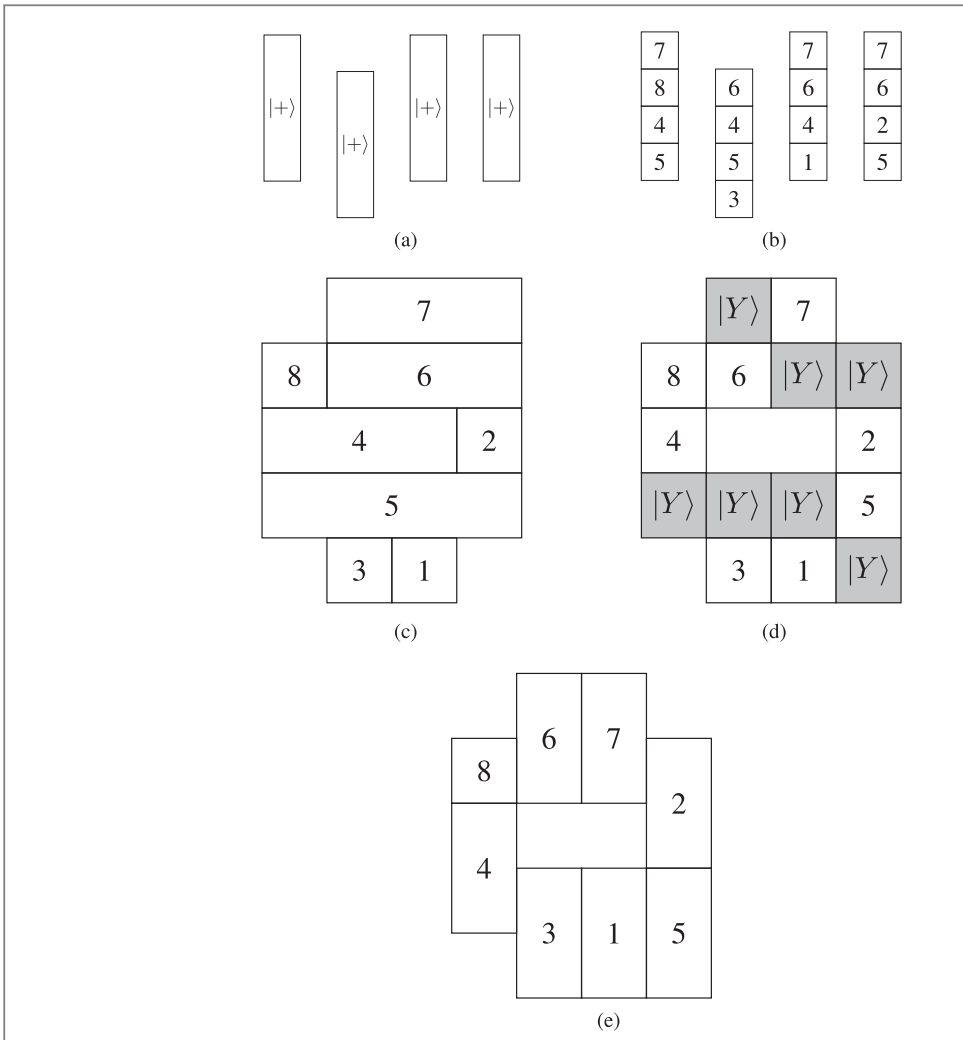


**Figure 14.** This figure shows the steps needed to perform lattice surgery. The numbers indicate which patch contributes to which qubit of the circuit in figure 13. Since merges and splits do not conserve the overall number of qubits, many patches might contribute to the same qubit at intermediate steps. (a) For each multi target CNOT a vertical block of surface code is initialized to $|+\rangle$. The height of these blocks is determined by the number of target qubits and how the circuit is optimized. (b) Each block is now separated into its qubits using smooth splits. (c) A rough merge connects the same target qubits of different CNOTs. (d) A shrink operation reduces the size of the each patch of unitary in order to make space for the $|Y\rangle$ states. (e) A smooth merge between a $|Y\rangle$-state and another qubit performs injection, such that the $Z$-gate is applied.

possible. However, higher levels of such concatenations need $|Y\rangle$-states calculated before. The transportation of these states complicates the geometry and further research is required.

### 8.2. Efficiency of the distillation

The total spatial requirements for one distillation run using the algorithm described above are given by $5 \times 4$ patches that encode a logical qubit each. If a rotated lattice is used, one will need $d^2$ data qubits, and $(d^2 - 1)$ syndrome qubits [33] for a distance $d$ surface code. This results in $2d^2$ physical qubits to leading order. The time requirements are given by $d$-cycles for the initialization of the states $|+\rangle$. Performing all the smooth splits in figure 14(b) will need $d$-cycles. The movement and merge operations need $d$-cycles each. Shrinking the qubits also needs $d$-cycles, while creating the injected $|Y\rangle$ states needs at least $d$-cycles. The final smooth merges take again $d$-cycles, such that the total time requirements sum to

$$t = 7d \qquad \text{cycles.}$$

This will give a total requirement of $2 \times 20 \times 7d^3 = 280d^3$ space–time volumes, if a rotated lattice is assumed (it should be noted that one step in time consists of two stabilizer rounds: $Z$-stabilizer and $X$-stabilizer). This performs worse than using the braiding description, which needs a space/time volume of $140d^3$ [28].

### 8.3. Stabilizer matrix calculation

The previously outlined algorithm can be calculated alternatively using the stabilizer matrix formulation with the rules presented before. In the beginning only four encoded qubits, in the state $|+\rangle$, exits, which can be

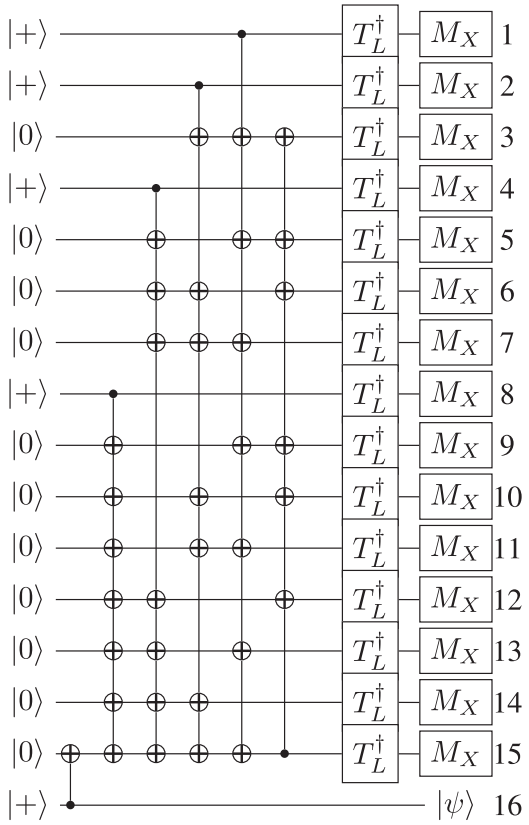


**Figure 15.** This circuit implements a Reed–Muller code which is used to distill $|A\rangle$. However, it is not yet possible to implement this circuit in lattice surgery since the last CNOT has a control qubit on a state that is targeted by other CNOTs. Using the classical algorithm provided in the supplementary, this circuit can be translated to a form that is implementable by lattice surgery.

represented as:

$$\begin{bmatrix} X & & & \\ & X & & \\ & & X & \\ & & & X \end{bmatrix}.$$

Using four smooth splits on each of these qubits will result in a four qubit GHZ-state for each

$$\left[\begin{array}{cccccccccccccccc} 7 & 8 & 4 & 5 & 6 & 4 & 5 & 3 & 7 & 6 & 4 & 1 & 7 & 6 & 2 & 5 \\ X & X & X & X & & & & & & & & & & & & \\ Z & Z & & & & & & & & & & & & & & \\ & Z & Z & & & & & & & & & & & & & \\ & & Z & Z & & & & & & & & & & & & \\ & & & & X & X & X & X & & & & & & & & \\ & & & & Z & Z & & & & & & & & & & \\ & & & & & Z & Z & & & & & & & & & \\ & & & & & & Z & Z & & & & & & & & \\ & & & & & & & & X & X & X & X & & & & \\ & & & & & & & & Z & Z & & & & & & \\ & & & & & & & & & Z & Z & & & & & \\ & & & & & & & & & & Z & Z & & & & \\ & & & & & & & & & & & & X & X & X & X \\ & & & & & & & & & & & & Z & Z & & \\ & & & & & & & & & & & & & Z & Z & \\ & & & & & & & & & & & & & & Z & Z \end{array}\right].$$

In this stabilizer matrix, the numbers correspond to the labeling that was used in figure 14(b). Now one has to proceed using the merge operations. The first merge is performed on the qubit labeled 7:

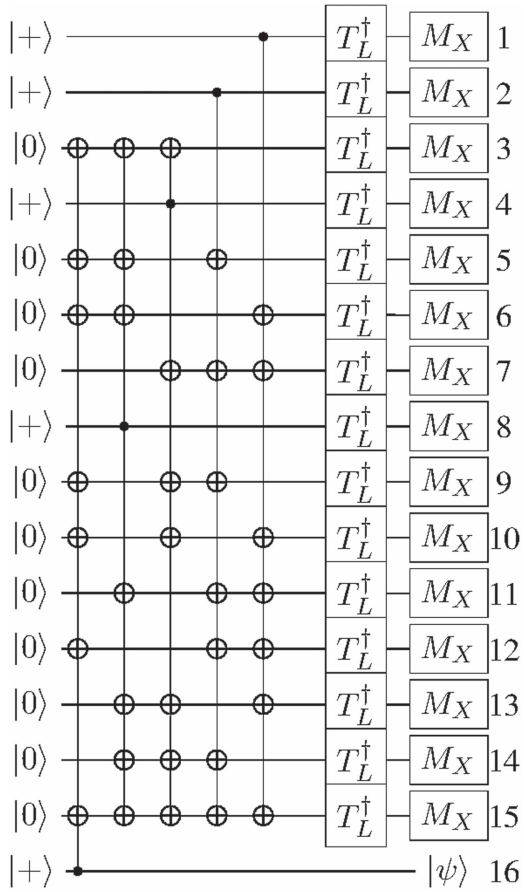


**Figure 16.** The last CNOT of the circuit in figure 15 has been eliminated resulting in an equivalent circuit that can be used with lattice surgery.

$$\left[\begin{array}{cccccccccccccc}
7 & 8 & 4 & 5 & 6 & 4 & 5 & 3 & 6 & 4 & 1 & 6 & 2 & 5 \\
X & X & X & X & & & & & & & & & & \\
Z & Z & & & & & & & & & Z & & Z & \\
 & Z & Z & & & & & & & & & & & \\
 & & Z & Z & & & & & & & & & & \\
 & & & & X & X & X & X & & & & & & \\
 & & & & Z & Z & & & & & & & & \\
 & & & & & Z & Z & & & & & & & \\
 & & & & & & Z & Z & & & & & & \\
X & & & & & & & & X & X & X & & & \\
 & & & & & & & & Z & Z & & & & \\
 & & & & & & & & & Z & Z & & & \\
X & & & & & & & & & & & X & X & X \\
 & & & & & & & & & & & Z & Z & \\
 & & & & & & & & & & & & Z & Z
\end{array}\right].$$

Using similar transformations, sorting the numbers and swapping the stabilizer rows, one obtains:

$$\left[\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
 & & X & X & X & X & & \\
 & X & & & X & & X & X \\
 & X & & & X & X & X & \\
 & & & X & X & & X & X \\
Z & Z & Z & & & Z & & \\
Z & Z & & & & & Z & Z \\
Z & & Z & Z & & & & Z \\
 & Z & Z & & Z & & & Z
\end{array}\right].$$

This stabilizer matrix describes the same state as the one given in [28], which was calculated using the circuit of figure 13.

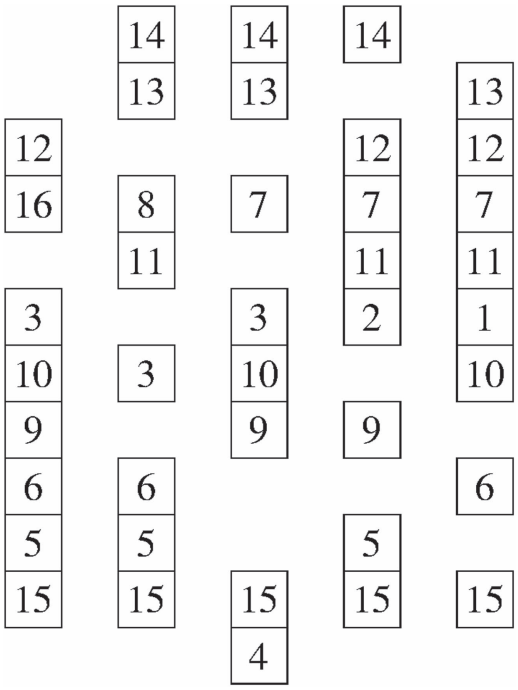


**Figure 17.** This shows the layout after the smooth splits and after their movement to their designated vertical positions. Each column represents one multi target CNOT with the qubits arranged in such a way that in the next step the merge operation can be performed. The empty spaces are initialized to 0, such that all merge operations can be applied in $d$ error-correcting cycles. As a side note, the difference of a $T^{\dagger}$-gate to a $T$-gate is only a change in corrective gates.

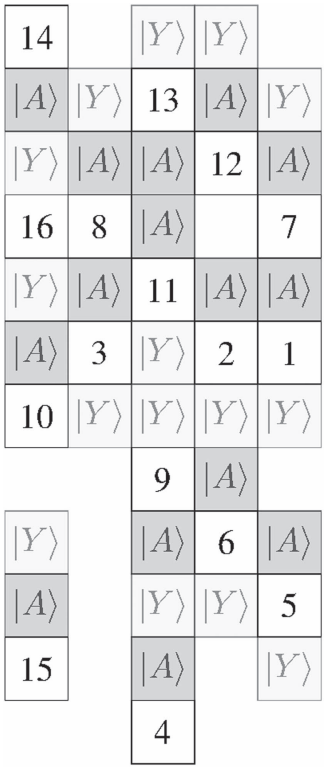


**Figure 18.** After the patches corresponding to the same qubits are merged in figure 17, excessive patches were shrunk, allowing space for state injection. Here, a possible layout with the correct positioning of $|A\rangle$-states and correctional $|Y\rangle$-states is given. In the next step a smooth merge is performed on every patch except the one labeled 16, which will store the output information.

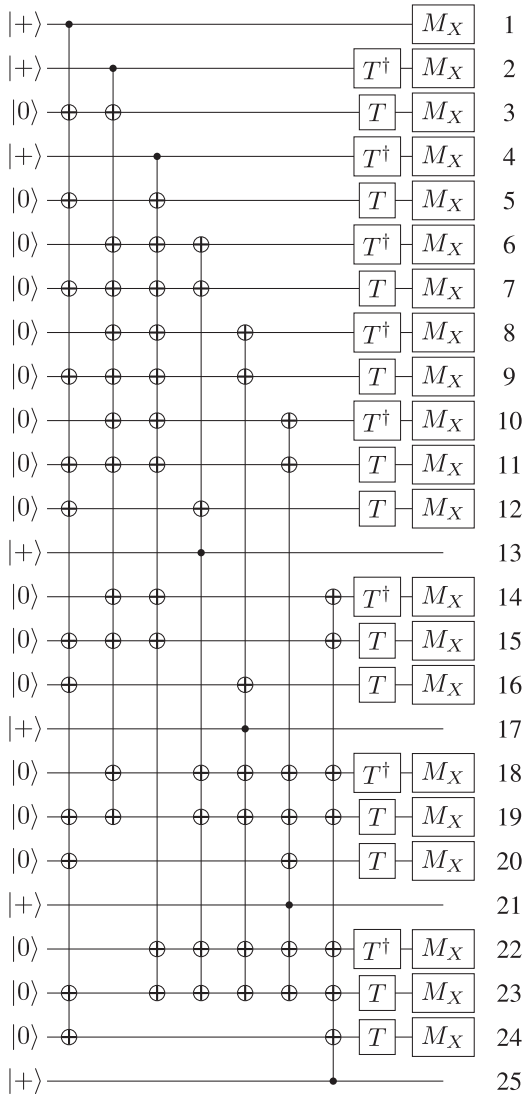


**Figure 19.** Circuit for the Bravyi–Haah $(3k + 8)$-to-$k$ distillation code for $k = 4$. The original circuit for this code was found in [31]. Using the algorithm given in the supplementary we translated it to this form, which can now be used for our translation to lattice surgery.

### 8.4. Reed–Muller code for $|A\rangle$-state-distillation

The distillation circuit for $|A\rangle$ is given by figure 15. One can notice that the last qubit needs to be permuted to the front of the circuit. This will be done using the algorithm included online [68]. The result of this transformation is shown in figure 16. After the translated circuit is obtained, it can be implemented using lattice surgery with the same steps as before. One choice for the layout of the patches is shown in figure 17 and the locations to inject $|A\rangle$ with their eventual corrective $|Y\rangle$ states are shown in figure 18. One can see that there are many *empty* regions that exist during the preparation of the entangled state, which we anticipate can be optimized further. The total spatial requirements for this circuit are given by 60 encoded regions. The time complexity in this circuit depends on how often an erroneous $T$ has been applied. If corrective $P$-gates have to be applied, the total time effort is higher than for the $P$-gate. Two additional steps to inject and merge corrective $|Y\rangle$-states are needed, giving a space–time volume of $1080d^3$. This compares with a space/time volume of $1500d^3$ for the braid-based logic [28].

### 8.5. Bravyi–Haah code for $|A\rangle$-state distillation

Another very promising class of distillation methods was introduced by Bravyi and Haah [38]. These are based on triorthogonal stabilizer matrices. In [31] an exemplary circuit that fulfills the triorthogonal requirement was already translated to the braiding framework. We use the same example, namely a $(3k + 8)$-to-$k$ distillation code for $|A\rangle$ with $k = 4$ and compare our estimates with those for braiding. Our translation again requires a circuit given in the inverse ICM format, whose CNOTs are then merged to as few as possible multi-target CNOTs. The result of this translation is given in figure 19. This circuit can now be translated to lattice surgery

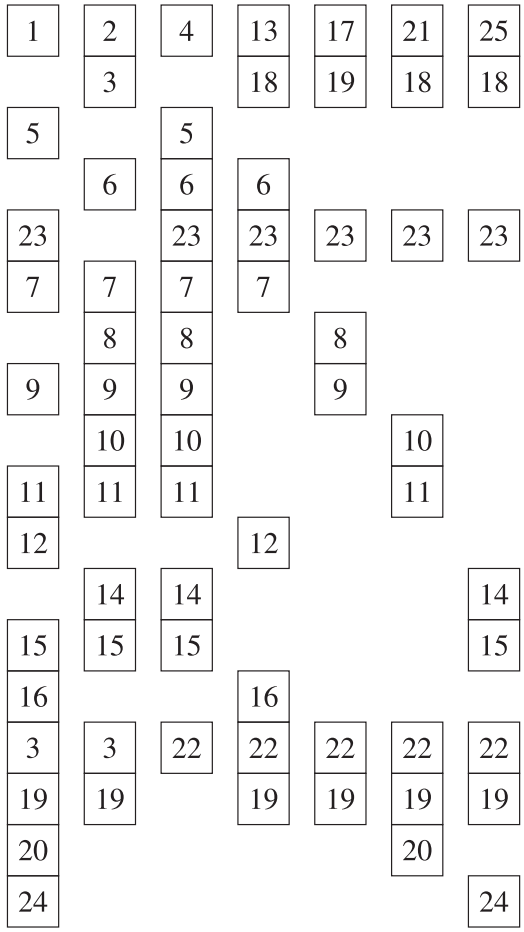


**Figure 20.** Continuing from the circuit in figure 19, we translate each multi-target CNOT into a row of surface code patches using smooth splits. The placement of these patches was again optimized manually.

using a new row for each multi-target CNOT. We optimized the layout of patches manually and obtained a space requirement of $7 \times 18$ patches of surface code. The time requirements do not change compared to the Reed–Muller code, such that this circuit needs $7d$ cycles for optimal performance, where no corrective $P$-gates are needed, and $9d$ cycles for a worst case. This calculates to a worst-case space–time volume of $2268d^3$ in lattice surgery. The braiding implementation of this circuit performs worse with a space–time volume of $4688d^3$. This is a good result; however, further research is required to obtain the scaling with $k$ for lattice surgery. In braiding, an efficient packing for arbitrary $k$ has already been found, whereas here we only performed manual optimization for this specific case.

## 9. Conclusion

In this paper we have provided a method for compiling a fault-tolerant quantum circuit for a surface code quantum computer based on lattice surgery protocols. Using the natural operations of the lattice surgery model and a specific representation of a compatible, fault-tolerant circuit, we show via stabilizers how the Clifford part of the circuit can be directly mapped to lattice surgery protocols in the surface code. Further work is required to optimize the arrangement and movement of encoded regions in the computer to efficiently realize any given circuit. Examples of state-distillation circuits were given, which have a comparable or better space/time volume than braid-based circuit implementations and with further optimization we expect this to decrease further. In light of recent results [45–47] on improved state-distillation procedures, further analysis should be conducted on how the resource cost changes with them applied to the lattice surgery model and with more efficient encodings now developed for the surface code [48, 49], qubit resources for an arbitrary algorithm will further decrease.

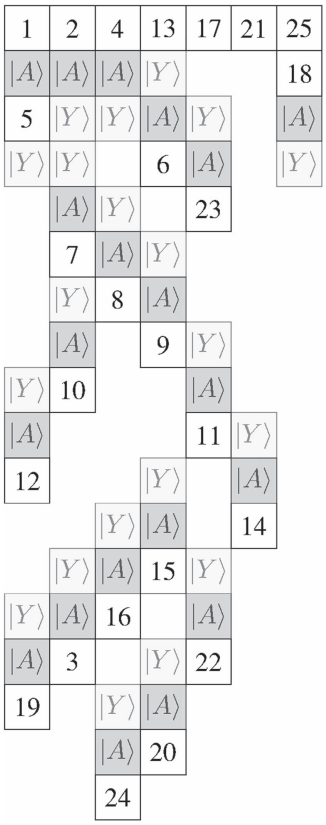


**Figure 21.** After rough merges of the same numbered patches and a shrink operation we provide a placement that allows for both the application of the $T$ or $T^{\dagger}$ gates and their corrective $P$ gates.

## Acknowledgments

DH is supported by the RIKEN IPA program. SJD acknowledges support from the JSPS Grant-in-aid for Challenging Exploratory Research and and the JST ImPact project, Japan. FN was partially supported by the RIKEN iTHES Project, the MURI Center for Dynamic Magneto-Optics, the IMPACT program of JST, CREST, a Grant-in-Aid for Scientific Research (A), and a grant from the John Templeton Foundation.

## Additional information

Source code for this work can be found at https://github.com/herr-d/LS_translation

## References

[1] Gaitan F 2008 *Quantum Error Correction and Fault Tolerant Quantum Computing* (Boca Raton, FL: CRC Press)
[2] Lidar D A and Brun T A (ed) 2013 *Quantum Error Correction* (Cambridge: Cambridge University Press)
[3] Gottesman D 2010 An Introduction to Quantum Error Correction and Fault-Tolerant Quantum Computation, *Quantum Information Science and Its Contributions to Mathematics, Proceedings of Symposia in Applied Mathematics* 68 13-58 (Providence, RI: American Mathematics Society)
[4] Devitt S J, Stephens A M, Munro W J and Nemoto K 2013 Requirements for fault-tolerant factoring on an atom-optics quantum computer *Nat. Commun.* **4** 2524
[5] Ahsan M, Van Meter R and Kim J 2016 Designing a million-qubit quantum computer using resource performance simulator *JETC* **12** 39
[6] Li Y, Humphreys P C, Mendoza G J and Benjamin S C 2015 Resource costs for fault-tolerant linear optical quantum computing *Phys. Rev.* X **5** 041007
[7] Kitaev A Y 2003 Fault-tolerant quantum computation by anyons *Ann. Phys., NY* **303** 2–30
[8] Dennis E, Kitaev A, Landahl A and Preskill J 2002 Topological quantum memory *J. Math. Phys.* **43** 4452
[9] Fowler A G, Stephens A M and Groszkowski P 2009 High threshold universal quantum computation on the surface code *Phys. Rev.* A **80** 052312
[10] Fowler A G, Mariantoni M, Martinis J M and Cleland A N 2012 Surface codes: towards practical large-scale quantum computation *Phys. Rev.* A **86** 032324

[11] Raussendorf R, Harrington J and Goyal K 2007 Topological fault-tolerance in cluster state quantum computation *New J. Phys.* **9** 199
[12] Bombin H and Martin-Delgato M A 2007 Optimal resources for topological 2D stabilizer codes: comparative study *Phys. Rev.* A **76** 012305
[13] Wang D S, Fowler A G, Stephens A M and Hollenberg L C L 2010 Threshold error rates for the toric and surface codes *Quantum Inf. Comput.* **10** 456
[14] Bombin H, Andrist R S, Ohzeki M, Katzgraber H G and Martin-Delgato M A 2012 Strong resilience of topological codes to depolarization *Phys. Rev.* X **2** 021004
[15] Stephens A M 2014 Fault-tolerant thresholds for quantum error correction with the surface code *Phys. Rev.* A **89** 022321
[16] Fujii K 2015 *Quantum Computation with Topological Codes: From Qubits to Topological Fault-Tolerance* (Berlin: Springer)
[17] Jones N C, Van Meter R, Fowler A G, McMahon P L, Kim J, Ladd T D and Yamamoto Y 2012 A layered architecture for quantum computing using quantum dots *Phys. Rev.* X **2** 031007
[18] DiVincenzo D P 2009 Fault-tolerant architectures for superconducting qubits *Phys. Scr.* **T137** 014020
[19] Monroe C, Raussendorf R, Ruthven A, Brown K R, Maunz P, Duan L-M and Kim J 2014 Large scale modular quantum computer architecture with atomic memory and photonic interconnects *Phys. Rev.* A **89** 022317
[20] Nemoto K, Trupke M, Devitt S J, Stephens A M, Buczak K, Nobauer T, Everitt M S, Schmiedmayer J and Munro W J 2013 Photonic architecture for scalable quantum information processing in NV-diamond *Phys. Rev.* X **4** 031022
[21] Lekitsch B, Weidt S, Fowler A G, Mølmer K, Devitt S J, Wunderlich C and Hensinger W K 2015 Blueprint for a microwave ion trap quantum computer arXiv:1508.00420
[22] Hill C D, Peretz E, Hile S J, House M G, Fuechsle M, Rogge S, Simmons M Y and Hollenberg L C L 2015 A surface code quantum computer in silicon *Sci. Adv.* **1** e1500707
[23] Devitt S J, Fowler A G, Tilma T, Munro W J and Nemoto K 2010 Classical processing requirements for a topological quantum computing system *Int. J. Quantum Inf.* **8** 1–27
[24] Duclos-Cianci G and Poulin D 2010 Fast decoders for topological quantum codes *Phys. Rev. Lett.* **104** 050504
[25] Fowler A G and Paetznick A 2013 Quantum circuit optimization by topological compaction in the surface code arXiv:1304.2807
[26] Devitt S J 2014 Classical control of large-scale quantum computers *RC2014* (*Springer Lecture Notes on Computer Science (LNCS)* vol 8507) pp 26–39
[27] Fowler A G 2015 Minimum weight perfect matching of fault-tolerant topological quantum error correction in average O(1) parallel time *Quantum Inf. Comput.* **15** 145–58
[28] Fowler A G and Devitt S J 2012 A bridge to lower overhead quantum computation arXiv:1209.0510
[29] Paler A, Devitt S J and Fowler A G 2016 Synthesis of arbitrary quantum circuits to topological assembly *Sci. Rep.* **6** 30600
[30] Paler A, Polian I, Nemoto K and Devitt S J 2015 A compiler for fault-tolerant high level quantum circuits arxiv:1509.02004
[31] Fowler A G, Devitt S J and Jones C 2013 Surface code implementation of block code state distillation *Sci. Rep.* **3** 1939
[32] Bravyi S B and Kitaev A Y 2001 Quantum codes on a lattice with boundary *Quantum Comput. Comput.* **2** 43
[33] Horsman C, Fowler A G, Devitt S J and Van Meter R 2012 Surface code quantum computing by lattice surgery *New J. Phys.* **14** 123011
[34] Bravyi S B and Kitaev A Y 2005 Universal quantum computation with ideal clifford gates and noisy ancillas *Phys. Rev.* A **71** 022316
[35] Calderbank A R and Shor P W 1996 Good quantum error-correcting codes exist *Phys. Rev.* A **54** 1098–105
[36] Nielsen M A and Chuang I L 2000 *Quantum Computation and Quantum Information* (Cambridge: Cambridge University Press)
[37] Meier A M, Eastin B and Knill E 2013 Magic-state distillation with the four-qubit code *Quant. Inf. Comp.* **13** 195–209
[38] Bravyi S and Haah J 2012 Magic-state distillation with low overhead *Phys. Rev.* A **86** 052329
[39] Jones C 2013 Multilevel distillation of magic states for quantum computing *Phys. Rev.* A **87** 042305
[40] Paler A and Devitt S J 2016 Verification of topological quantum circuits (in preparation)
[41] Browne D E and Rudolph T 2004 Resource-efficient linear optical quantum computation *Phys. Rev. Lett.* **95** 010501
[42] Beenakker C W J, DiVincenzo D P, Emary C and Kindermann M 2004 Charge detection enables free-electron quantum computation *Phys. Rev. Lett.* **93** 020501
[43] Raussendorf R and Briegel H J 2001 A one-way quantum computer *Phys. Rev. Lett.* **86** 5188–91
[44] Briegel H J, Browne D E, Dür W, Raussendorf R and Van den Nest M 2009 Measurement-based quantum computation *Nat. Phys.* **5** 19–26
[45] O'Gorman J and Campbell E T 2016 Quantum computation with realistic magic state factories arXiv:1605.07197
[46] Campbell E T and Howard M 2016 A unified framework for magic state distillation and multi-qubit gate-synthesis with reduced resource cost arXiv:1606.01904
[47] Campbell E T and O'Gorman J 2016 An efficient magic state approach to small angle rotations *Quantum Sci. Technol.* **1** 015007
[48] Nagayama S, Satoh T and Van Meter R 2016 State injection, lattice surgery and dense packing of deformation-based surface code arxiv:1606.06072
[49] Delfosse N, Iyer P and Poulin D 2016 Generalized surface codes and packing of logical qubits arxiv:1606.07116
[50] Yao X *et al* 2012 Experimental demonstration of topological error correction *Nature* **482** 489–94
[51] Barends R *et al* 2014 Superconducting quantum circuits at the surface code threshold for fault tolerance *Nature* **508** 500–3
[52] Hornibrook J M *et al* 2015 Cryogenic control architecture for large-scale quantum computing *Phys. Rev. Appl.* **3** 024010
[53] Fowler A G, Whiteside A C and Hollenberg L C L 2012 Towards practical classical processing for the surface code: timing analysis *Phys. Rev.* A **86** 042313
[54] Kliuchnikov V, Maslov D and Mosca M 2012 Fast and efficient exact synthesis of single qubit unitaries generated by clifford and T-gates *Quantum Info. Comput.* **13** 607–30
[55] Giles B and Selinger P 2013 Exact synthesis of multiqubit clifford + *T* circuits *Phys. Rev.* A **87** 032332
[56] Bocharov A, Roetteler M and Svore K M 2015 Efficient synthesis of probabilistic quantum circuits with fallback *Phys. Rev.* A **91** 052317
[57] Ross N J and Selinger P 2014 Optimal ancilla-free clifford + *T* approximation of Z-rotations arXiv:1403.2975
[58] Häner T, Steiger D S, Svore K and Troyer M 2016 A software methodology for compiling quantum programs arXiv:1604.01401
[59] Paler A, Devitt S J, Nemoto K and Polian I 2014 Mapping of topological quantum circuits to physical hardware *Sci. Rep.* **4** 4657
[60] Hordern E 1986 *Sliding Piece Puzzles* (Oxford: Oxford University Press)
[61] Fowler A G 2012 Time optimal quantum computation arxiv:1210.4626
[62] Gottesman D 1999 The Heisenberg Representation of Quantum Computers *Proc. 22nd Int. Colloquium on Group Theoretical Methods in Physics* ed S P Corney *et al* (Cambridge, MA: International Press) pp 32–43

[63] Aliferis P 2007 Level Reduction and the quantum threshold theorem *PhD Thesis* Caltech
[64] Jones N C 2013 Logic synthesis for fault-tolerant quantum computers *PhD Thesis* Stanford University
[65] Fowler A G 2012 Low overhead surface code logical hadamard *Quantum Inf. Comput.* **12** 970
[66] Łodyga J, Mazurek P, Grudka A and Horodecki M 2015 Simple scheme for encoding and decoding a qubit in unknown state for various topological codes *Sci. Rep.* **5** 8975
[67] Li Y 2015 A magic state's fidelity can be superior to the operations that created it *New J. Phys.* **17** 023037
[68] Gitlab Repository to this work https://github.com/herr-d/LS_translation