Quantum Information Processing (2024) 23:190
https://doi.org/10.1007/s11128-024-04389-8

Fault-tolerant quantum computation using low-cost joint measurements

Yujin Kang[1] · Jonghyun Lee[1] · Jinyoung Ha[1] · Jun Heo[1]

Received: 1 November 2023 / Accepted: 12 April 2024 / Published online: 14 May 2024

Abstract
We introduce a new method to implement joint measurements using a 4-qubit twist defect on a rotated surface code. The proposed method enables us to perform logical S (Phase gate), T ($\pi/8$ gate), and H (Hadamard) with low overhead. Combined with other universal quantum gates, we can implement fault-tolerant quantum computation at the lattice surgery level beyond the gate level while saving considerable resources. We compare our method with previous methods using benchmark circuits by calculating the space and time costs. The proposed method requires additional lines of physical qubits for each encoded patch. Although it slightly increases the space cost for logical H compared to the previous work, it reduces the time cost. In addition, the proposed method decreases the space cost and time cost by introducing a 4-qubit twist defect for logical S and T. Therefore, the overall space-time cost is reduced.

Keywords Fault-tolerant quantum computation · Quantum error correction code · Surface code · Lattice surgery · Logical operation

1 Introduction

To protect quantum information from errors arising from the unstable characteristics of physical qubits, a quantum error correction code (QECC) is required [1, 2]. A surface code is a type of topological code with a high threshold and can be implemented

✉ Jun Heo
junheo@korea.ac.kr

Yujin Kang
yujin20@korea.ac.kr

Jonghyun Lee
ljh0523@korea.ac.kr

Jinyoung Ha
ksuwer@korea.ac.kr

[1] School of Electrical Engineering, Korea University, Seoul 02841, Republic of Korea

using operations between adjacent qubits in a two-dimensional space. Therefore, it is a promising technology for realistic architectures such as superconducting qubits, optical lattices, ion traps, quantum dots, and nitrogen-vacancy (NV) centers, which can perform physical operations between adjacent qubits [3–8]. In addition, the high threshold of the surface code is maintained even when a gate error of 0.8% to 1.4% occurs [4]. Although there is a coherent error, the threshold is undamaged as the code distance is increased [9]. Therefore, this study targets a fault-tolerant quantum system based on a surface code, especially a rotated surface code requiring fewer physical qubits per logical qubit [10].

Once a quantum error correction code has been determined, we can implement a quantum system that performs any quantum operation at a logical level using a universal quantum gate set on the encoded data. A universal quantum gate set is a set of quantum operations consisting of Clifford gates (Pauli, CNOT, Hadamard H, phase gate S) and a non-Clifford gate ($\pi/8$ gate T). Any unitary operation can be decomposed into the elements of a universal quantum gate set with arbitrary accuracy [11]. Therefore, a fault-tolerant quantum computation (FTQC) can be achieved if a fault-tolerant universal gate set can be implemented.

Pauli operations have zero cost in terms of time and space resources by manipulating the results of measurements in software [12]. Other logical operations in a universal quantum gate set can be performed using joint measurements that can be implemented by lattice surgery. In previous studies, a logical CNOT has been implemented using two joint measurements and one measurement on a single qubit [13–15]. However, the time cost of a logical CNOT can increase, depending on the logical qubit arrangement. In the case of the row-type architecture (r-arch) [16], the logical CNOT takes $2d$ cycles for error correction. In checkerboard-type (c-arch) and tile-based architectures (t-arch) [17], qubit movements are required to perform a logical CNOT between two distant qubits while increasing the time cost to $3d \sim 4d$ cycles.

For logical S (S_L) and T (T_L) gates, a magic state refined through a distillation process is needed [15, 18]. For the S_L gate, we execute a quantum teleportation circuit [12] or joint $Z_L Z_L$ measurement [19] with the magic state. Alternatively, we can implement S_L using a joint $Z_L Y_L$ measurement with logical $|0\rangle$ ($|0\rangle_L$) [20]. However, this approach requires two-patch-wide logical qubits. T_L, a non-Clifford gate, requires a distilled magic state and can be implemented using a quantum teleportation circuit [21] or a sequence of joint measurements [15]. For logical H (H_L), we must perform physical Hadamards transversally and lattice surgery with adjacent areas to return to the original position [14, 15]. Recently, circuit-level analysis for a similar method has been conducted [22]. A quantum teleportation circuit using lattice surgery with an additional logical qubit can remove these physical operations while increasing time and space costs [23].

After determining how to perform logical operations on the encoded information using joint measurements, the next step is to arrange the logical qubits. When we use surface code, logical qubits are counted in patches. Depending on how the stabilizers are constructed, a single patch may represent logical information with two or more degrees of freedom. The X and Z stabilizers of the surface code are shown in Fig. 1a. The numbers and blue arrows indicate the order of the physical CNOTs in the parity measurement circuit. Figure 1b shows a logical qubit with two degrees of freedom,

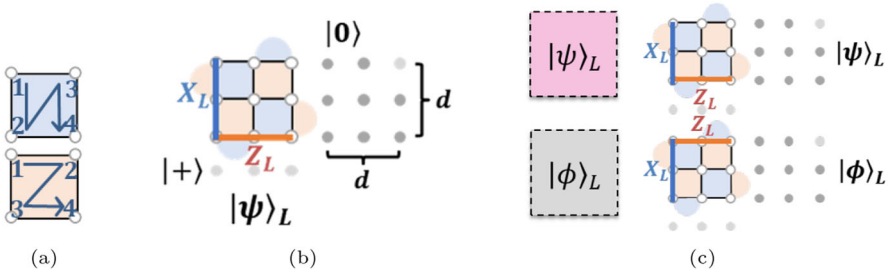


Fig. 1 **a** X and Z stabilizers. Each circle represents one qubit. The blue (or red) face represents the X (or Z) stabilizer. Each face includes a measurement qubit. The execution order of each stabilizer is determined in a zigzag manner [13]. The top-to-bottom sequence in the X stabilizers and the left-to-right sequence in the Z stabilizers are indicated by blue arrows. **b** Logical qubit encoded using rotated surface code with a distance of 3. The gray circles represent ancilla qubits for lattice surgery, and the white circles represent data qubits. The logical X (X_L) and logical Z (Z_L) operations are defined in a sequence of X or Z marked by a thick solid line. X_L is encoded by one of the Z boundaries, and Z_L is encoded by one of the X boundaries. This can be changed by H operations. **c** Patches encoded using the rotated surface code in (**b**). The pink and gray boxes represent the data and ancilla patches, respectively. The logical qubits on the right correspond to patches on the left (Color figure online)

and Fig. 1c shows two types of patches. One patch in which the user performs the desired operations is called a *data patch*, and the other patch consumed in the middle of the operation is called an *ancilla patch*. There are various logical qubit architectures, depending on how the data and ancilla patches are arranged, and each structure has different efficiencies in terms of time and space costs [16, 17]. In this study, we assume that an ancilla patch is placed above or below a data patch. Furthermore, we initially fix all patches' X and Z boundaries in the same direction so that logical operations can be performed in a certain direction. These assumptions are common to all architectures in [16, 17]. However, we add physical qubits between the patches, represented by the gray circles in Fig. 1b. The proposed joint measurement is described based on a rotated surface code, and a method for implementing a universal quantum gate set using the proposed techniques is presented. Because our method rotates the boundary when performing a H_L, it uses less time than previous methods without physical H operations. In addition, by presenting a method for performing a S_L without a magic state, we can efficiently perform S_L and T_L that requires an S correction after measurement.

The remainder of this paper is organized as follows. In Sect. 2, we review some of the operations proposed in previous studies that are applied to this paper. Therefore, we describe a previous work for implementing joint $X_L X_L$, $Z_L Z_L$, and $X_L Z_L$ measurements based on the rotated surface code in Sect. 2.1 and 2.2. Logical operations such as initialization, state injection, measurement, Pauli, and CNOT based on joint $X_L X_L$ and $Z_L Z_L$ measurements are also described in Sect. 2.3. New methods for implementing S_L and T_L using a joint $Z_L Y_L$ measurement and H_L using a boundary rotation are presented in Sect. 3. A joint $Z_L Y_L$ measurement based on a 4-qubit twist defect is proposed in Sect. 3.1. Sections 3.2 and 3.3 perform S_L and T_L using the proposed $Z_L Y_L$ measurement while consuming fewer resources than previous studies. In addition, Sect. 3.4 implements a H_L based on a boundary rotation and the transfor-

mation of the subsequent measurement with fewer space-time resources. In Sect. 4, the number of physical qubits (space cost), number of cycles (time cost), and product of the space and time cost (space-time cost) required for benchmarks are presented using logical operations in Sect. 2 and 3. The conclusions are discussed in Sect. 5.

2 Preliminary

2.1 Joint $X_L X_L$,$Z_L Z_L$ measurement

Lattice surgery is used to manipulate logical qubits by activating specific stabilizers or measuring certain physical qubits. It involves two techniques. *Merging* is one of the techniques that involves turning on stabilizers between two patches to unify them. This operation projects united patch onto one of the eigenspaces of the product of logical operations. The other is *splitting*, which involves measuring certain physical qubits of a patch and dividing it into two individual patches. Because this action makes some stabilizers in the cross-section inactive, the information on the unified patch is distributed to two separate patches.

When merging two qubits facing each other's Z boundary, the product of parity measurements for the newly activated stabilizers is equal to the product of the measurement outcome concerning $X_L X_L$. Therefore, Z boundary merging projects patch into one of the eigenspaces of $X_L X_L$, called joint $X_L X_L$ measurement. Likewise, X boundary merging is called joint $Z_L Z_L$ measurement. After merging, the two patches are unified and must be split to apply logical operations to the two individual patches or shrink them to return to their original size [14, 15, 23].

In addition, we can implement *patch extension* and *patch shrinkage* using joint $X_L X_L$ and $Z_L Z_L$ measurements. Performing the joint $Z_L Z_L$ measurement with $|+\rangle_L$ or the joint $X_L X_L$ measurement with $|0\rangle_L$ leaves the information of the data patch unchanged, increasing the patch size, so-called *patch extension*. The code distance during the joint measurement is maintained, and the entire procedure takes $1d$ cycles. However, measuring an extended part in $X(Z)$ basis in the case of the joint $Z_L Z_L$ ($X_L X_L$) measurement results in *patch shrinkage*, which takes $0d$ cycles. Figure 2a, b shows a *patch extension* using the joint $Z_L Z_L$ measurement with $|+\rangle$, and a *patch shrinkage* by measuring in X basis is presented in Fig. 2c, d.

2.2 Joint $X_L Z_L$ measurement

We consider joint measurements facing the same boundaries, as described in the previous section. This section introduces a joint $X_L Z_L$ measurement for different boundaries. In [13] and [15], a joint $X_L Z_L$ measurement was performed using XZ stabilizers consisting of two X and Z operators. Figure 3 shows an example of the $X_L Z_L$ measurement for the two patches.

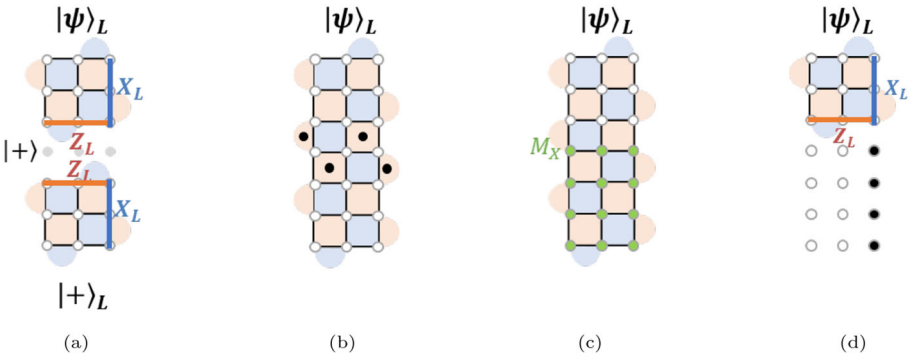


Fig. 2 An example of *patch extension* and *patch shrinkage*. **a** A data patch in an arbitrary quantum state $|\psi\rangle$ and an ancilla patch initialized in $|+\rangle$. $X_L = XXX$ and $Z_L = ZZZ$ are marked by bold solid lines. **b** The joint $Z_L Z_L$ measurement to extend the data patch in the size of two-patch-wide. Using the product of the stabilizers marked by black circles, we can obtain the outcome of the $Z_L Z_L$ measurement. **c** Measuring physical qubits colored in green on X basis to shrink the patch. **d** Compensation for logical X operators. The product of the measurement outcomes of qubits colored in black must be reflected in the X_L after the shrinkage [15] (Color figure online)

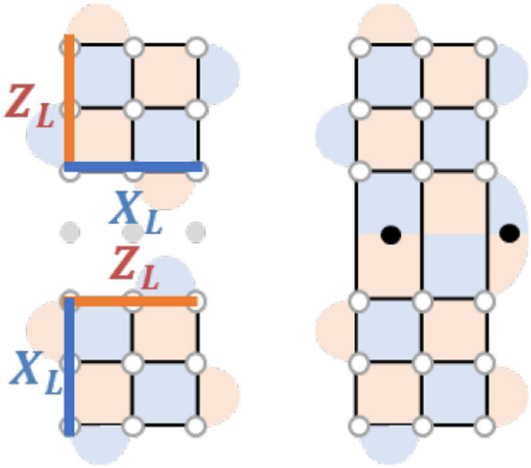


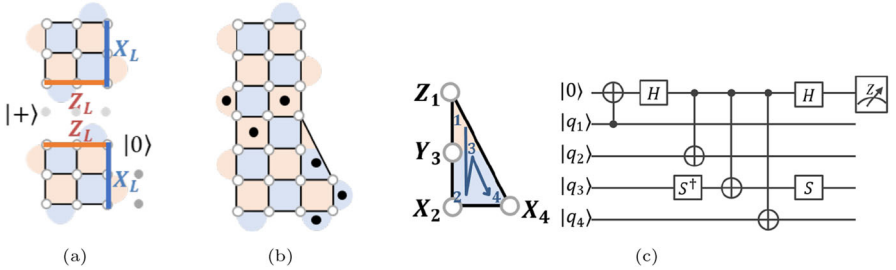
Fig. 3 Joint $X_L Z_L$ measurement of two patches. The product of the parity measurements marked by black circles is equivalent to the outcome of $X_L Z_L$ measurement

2.3 Initialization, state injection, measurement, Pauli operator, CNOT

We use the method presented in [14, 15, 20] for logical operations such as initialization, state injection, measurement, and CNOT. The initialization, state injection, and measurement take $0d$ cycles, whereas the CNOT requires two joint measurements and takes $2d$ cycles. However, when considering the logical qubit architecture, logical CNOT requires additional logical SWAP or merging with surrounding areas [16, 17], resulting in longer surface code cycles. Pauli operations are processed in the software using a commutation relationship with other operations [12, 24]. Thus, $0d$ cycles are required. Table 1 summarizes the logical operations and the cost in previous studies.

Table 1 The cost of logical operations in previous work

Operation	Time cost	Space cost	References
Initialization, state injection, measurement	$0d$	$1d^2$	[14]
Patch shrinkage	$0d$	$1d^2$	[20]
Patch extension	$1d$	$2d^2$	[20]
Pauli	$0d$	$1d^2$	[12, 24]
CNOT(r-arch)	$2d$	$3d^2$	[14–16]
CNOT(c-arch)	$3d$	$3d^2$	[17]
CNOT(t-arch)	$4d$	$3d^2$	[17]



(a) (b) (c)

Fig. 4 The proposed $Z_L Y_L$ measurement process. **a** A data patch in an arbitrary quantum state. The light gray circles represent ancilla qubits initialized in $|+\rangle$. X stabilizers do not affect light gray ancilla qubits because they are already in a $+1$ eigenspace of X stabilizers. Dark gray circles are ancilla qubits initialized in $|0\rangle$. Ancilla qubits in $|0\rangle$ can ignore Z stabilizers for the same reason as ancilla qubits in $|+\rangle$. **b** A $Z_L Y_L$ measurement between two patches. The product of stabilizers marked in black circles yields the outcome of $Z_L Y_L$ measurement. **c** A 4-qubit twist defect and corresponding circuit for parity measurement. In the circuit, the first qubit is a syndrome qubit containing the parity check's result

3 Logical operations using joint measurements

3.1 Joint $Z_L Y_L$ measurement

In [13], a 5-qubit twist defect has been introduced for a logical Y (Y_L) measurement that requires two-patch-wide logical qubits. This study proposes a $Z_L Y_L$ measurement between one-patch-wide logical qubits using a 4-qubit twist defect. A 4-qubit twist defect can be implemented in eight steps, unlike a 5-qubit defect. Consequently, the proposed defect does not increase the measurement-failure probability and the total time cost after scheduling logical operations. In addition, the effective code distance is not reduced in the minimum-weight-perfect-matching (MWPM) decoder [25]. From the newly active stabilizers, the Z_L of the first qubit and Y_L of the second qubit can be obtained to implement the $Z_L Y_L$ measurement. Figure 4 shows the $Z_L Y_L$ measurement process for the rotated surface code (SC-17) [10] at a distance of 3. When the distance is 3, six stabilizers are newly turned on, as shown in Fig. 4b. Three of them are Z stabilizers, two are X stabilizers, and the other is a mixed stabilizer that includes X, Y, and Z, called a 4-qubit twist defect. The product of the Z stabilizers and a mixed

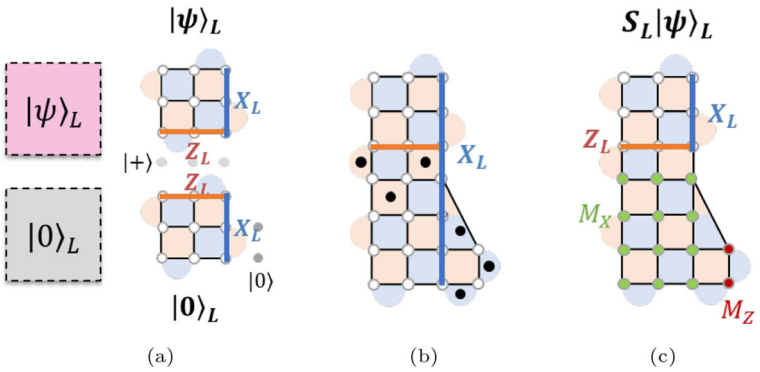


Fig. 5 The proposed S_L process. **a** A data patch (pink box) in an arbitrary state $|\psi\rangle_L$ and an ancilla patch (gray box) in $|0\rangle_L$. Corresponding logical qubits encoded by SC-17 are placed next to the boxes. The initialization of two patches followed by lattice surgery requires $0d$ cycles **b** $Z_L Y_L$ measurement between $|\psi\rangle_L$ and $|0\rangle_L$. The product of stabilizers marked in black circles is equivalent to the outcome of the $Z_L Y_L$ measurement. **c** Patch shrinkage to return the patch to the original size. Qubits marked in green or red are measured in X or Z basis. The measurement outcomes of qubits constituting the X_L in (**b**) are used to calibrate the reduced X_L (Color figure online)

stabilizer produces the $Z_L Z_L$ of the two logical qubits, and multiplication by the X stabilizers results in the logical X of the second qubit, that is, the product of the new active stabilizers is equal to the product of the Z_L of the first logical qubit and the Y_L of the second logical qubit. A quantum circuit for parity measurement of the mixed stabilizer is shown in Fig. 4c. The execution order of the gates is determined to be Z_1-X_2-Y_3-X_4, considering the surrounding stabilizers. The distance between the logical X and Z operators is maintained at d. We present a 3D space-time figure where the logical operators are supported in Appendix A. The proof of the $Z_L Y_L$ measurement using the stabilizer description is presented in Appendix B.

3.2 S_L gate

Logical S can be implemented using Hadamards and CNOTs with magic state $|Y\rangle_L$ [12] or a joint $Z_L Z_L$ measurement with $|Y\rangle_L$ [19]. The magic state $|Y\rangle_L$ can be expressed as

$$|Y\rangle_L = \frac{1}{\sqrt{2}}(|0\rangle_L + i\,|1\rangle_L). \tag{1}$$

This can also be implemented using a joint $Z_L Y_L$ measurement with $|0\rangle_L$ [20]. In particular, authors in [20] used two-patch-wide logical qubits to perform a joint $Z_L Y_L$ measurement. However, we propose a method for one-patch-wide logical qubits. We utilize the data patch shown in Fig. 1, but only $1d^2 + 2d$ physical qubits are used per patch. In Fig. 5, one data patch requires an ancilla patch in $|0\rangle_L$ to perform an S_L operation. The result of the joint $Z_L Y_L$ measurement between the two patches is equivalent to performing an S_L on the data patch, and patch shrinkage is applied by measuring some qubits to return to the original size. Therefore, as shown in Fig. 5c, the qubits marked in green are measured in the X basis, and the qubits marked in red

Table 2 Comparison of the implementation of S_L operation

Method	MSD $\|Y\rangle_L$	Time cost	Space cost (data+ancilla patches)	References
Hadamards, CNOTs with $\|Y\rangle_L$	◯	$14d$	$2 \times 1d^2$	[12]
$Z_L Z_L$ measure with $\|Y\rangle_L$	◯	$1d$	$2 \times 1d^2$	[19]
$Z_L Y_L$ measure with $\|0\rangle_L$ (5-qubit defect)	×	$1d$	$2 \times (2d^2 - 2d)$	[20]
$Z_L Y_L$ measure with $\|0\rangle_L$ (4-qubit defect)	×	$1d$	$2 \times (1d^2 + 2d)$	Proposed

are measured in the Z basis. Pauli correction is also required according to the results of those measurements. The entire operation takes $1d$ cycles.

The results of comparing the proposed method with the previous studies are presented in Table 2. The space cost is calculated by considering all the consumed ancilla patches and the data patch where S_L is performed. The first method uses logical Hadamards and CNOTs, which requires a considerable amount of time. Because all methods except the first perform the joint measurement once, they take $1d$ cycles to complete the error correction. Although the second method takes $1d$ cycles to finish the error correction, magic state distillation (MSD), which requires many resources in fault-tolerant quantum computation, is required to prepare the $|Y\rangle_L$ [12, 18, 26]. The third method does not require distilled $|Y\rangle_L$. However, it uses two-patch-wide logical qubits and requires the largest number of physical qubits (space cost). The proposed method uses a 4-qubit twist defect and does not require a magic state distillation for $|Y\rangle_L$. Therefore, only a small number of qubits are required.

3.3 T_L gate

Since the T_L is a non-Clifford operation, magic state $|A\rangle_L$ is required [18, 27–29]. The magic state $|A\rangle_L$ can be expressed as

$$|A\rangle_L = \frac{1}{\sqrt{2}}(|0\rangle_L + e^{\frac{i\pi}{4}}|1\rangle_L). \tag{2}$$

Previous studies have implemented T_L using logical CNOT and S_L [21] or a joint $Z_L Z_L$ measurement with the magic state [15, 30]. In this study, we present a method that consumes less time and space resources by applying the S_L proposed in the previous section. Our method is based on a circuit that performs T_L using $Z_L Z_L$ and $Z_L Y_L$ measurements with $|A\rangle_L$ and $|0\rangle_L$ [20]. Figure 6 shows the process of $Z_L Z_L$ and $Z_L Y_L$ measurements with $|A\rangle_L$ and $|0\rangle_L$ to perform T_L on an arbitrary state $|\psi\rangle_L$. All stabilizers for $Z_L Z_L$ and $Z_L Y_L$ measurements are turned on simultaneously in one surface code cycle, as shown in Fig. 6b. Because the initialization of the $|0\rangle_L$ accompanied by lattice surgery requires $0d$ cycles, the total time cost is still $1d$ cycles. The result of the $Z_L Z_L$ (or $Z_L Y_L$) measurement is the product of the parity measurements of the stabilizers, which are marked by blue (or black) circles in Fig. 6b. S_L correction

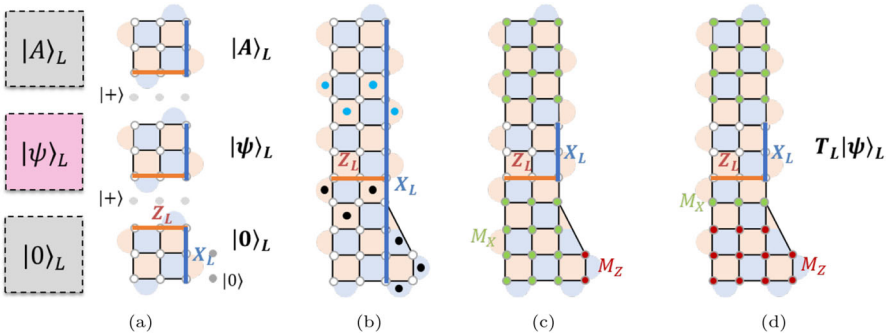


Fig. 6 The proposed T_L process. **a** A data patch (pink box) in an arbitrary state $|\psi\rangle_L$ and ancilla patches(gray box) in $|A\rangle_L$ and $|0\rangle_L$. Corresponding logical qubits are placed next to the boxes. **b** $Z_L Z_L$ and $Z_L Y_L$ measurements on $|\psi\rangle_L$ with $|A\rangle_L$ and $|0\rangle_L$. The product of stabilizers marked in blue(black) circles is equivalent to the outcome of the $Z_L Z_L (Z_L Y_L)$ measurement. **c** S_L correction by measuring $|0\rangle_L$ patch in X basis. **d** Measuring $|0\rangle_L$ patch in Z basis to cancel the S_L correction. The measurement outcomes of qubits constituting the logical X in (**b**) are used to calibrate the reduced logical X (Color figure online)

is required according to the $Z_L Z_L$ and $Z_L Y_L$ measurement results. Using the conditional S_L circuit shown in [20], if an S_L correction is required, we measure the $|0\rangle_L$ patch in X basis, as shown in Fig. 6c. If a correction is unnecessary, we measure the patch in Z basis, as shown in Fig. 6b.

Table 3 summarizes the time and space cost of T_L operations. The first method requires the most time resources because it uses a logical CNOT and a S_L. The second method takes $2d$ cycles because of a joint measurement followed by a S_L. Since the third method and the proposed method need only one joint measurement, they require $1d$ cycles, which is the shortest of all. The proposed method uses $2d$ more qubits per patch, but still less than the third method.

3.4 H_L gate

In previous studies, a H_L has been implemented using transversal physical Hadamards [14] or a quantum teleportation circuit without physical Hadamards [23]. In [14], the boundaries are not aligned in subsequent joint measurements because physical Hadamards change the stabilizers and boundaries. Therefore, rotation is performed by merging with extra space, as shown in Fig. 7. To reserve the extra space, the number of physical qubits per logical qubit is enlarged by as much as $2d^2$.

In [23], the authors utilized quantum teleportation with $|0\rangle_L$ to perform a H_L on a single data patch. Therefore, one ancilla patch in $|0\rangle_L$ is consumed, and the data to which H_L is applied moves on to the ancilla patch. For c-arch, t-arch[17], and r-arch[16], this method causes the H_L to be considered during routing. In addition, the time cost required to return the data to the data patch position increases. Consequently, this method increases not only the space cost but also the time cost.

Before proposing a H_L, changing the type of the boundaries is defined as a *boundary rotation*, which is presented in [20]. Figure 8 illustrates the execution sequence based in Fig. 1b. Before the boundary rotation (normal cycle), the X boundaries are composed

Table 3 Comparison of the implementation of T_L operation

Method	MSD $\|Y\rangle_L$	Time cost	Space cost (data+ancilla patches)	References
CNOT with $\|A\rangle_L$ and S_L correction	◯	$17d$	$3 \times 1d^2$	[21]
$Z_L Z_L$ measure with $\|A\rangle_L$ and S_L correction	◯	$2d$	$3 \times 1d^2$	[15, 30]
$Z_L Z_L$, $Z_L Y_L$ measure with $\|A\rangle_L$, $\|0\rangle_L$(5-qubit defect)	×	$1d$	$2 \times (2d^2 - 2d) + 1d^2$	[20]
$Z_L Z_L$, $Z_L Y_L$ measure with $\|A\rangle_L$, $\|0\rangle_L$(4-qubit defect)	×	$1d$	$3 \times (1d^2 + 2d)$	Proposed

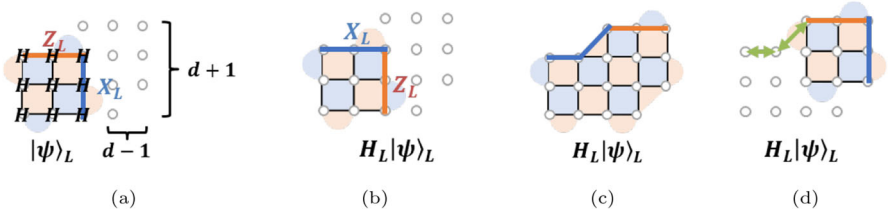


Fig. 7 **a** A data patch in an arbitrary quantum state. A H_L is performed using transversal Hadamard operations on physical qubits followed by $1d$ rounds of error correction. **b** After transversal Hadamard operations, X stabilizers turn into Z stabilizers and vice versa. It appears that the whole patch rotated by 90°. **c** The patch is merged with surrounding areas, while the distance of logical qubit is maintained, taking $1d$ cycles. **d** The patch is shrunk, leaving the shape consistent with (**a**). After the shrinkage, physical qubits are initialized and physical swap operations move the whole patch to its original position, taking $1d$ cycles each

of X stabilizers above and below the patch, and the Z boundaries are composed of Z stabilizers on the left and right sides. After the X boundary on the top is expanded to the right side, we activate adjacent Z stabilizers instead of X stabilizers at the top boundary as though we expand the Z boundary on the left. Then, some qubits are measured in X basis to reduce the patch, and it is expanded to the left while increasing only Z boundaries. Again, some qubits are measured in X basis, and the patch is reduced to its original size. When a boundary rotation is performed, all stabilizers on the boundaries are changed after $3d$ cycles. The boundaries are rotated by 90°, leading to rotations of the X_L and Z_L. The stabilizer changes simultaneously within each stage, and the distance and degrees of freedom remain the same; thus, the original information is not damaged. However, the logical operator must be corrected based on

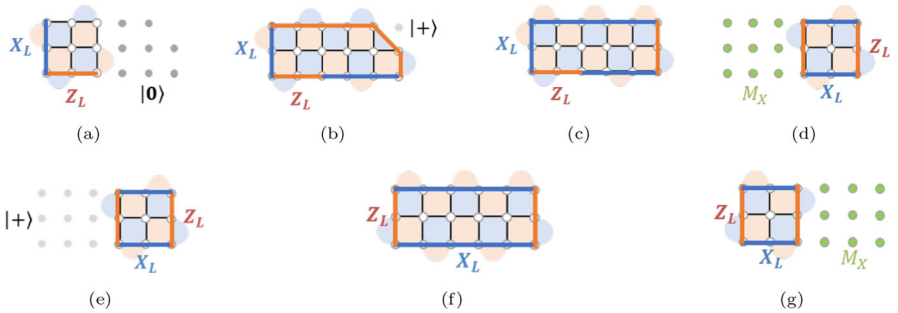


Fig. 8 Boundary rotation process. **a** A boundary rotation starting from Fig. 1b. Ancilla qubits colored in dark gray are initialized in $|0\rangle$. **b** Expanding X boundary on the top to the right. The minimum weight of logical operators remains $d = 3$. After expansion, $1d$ rounds of surface code cycles are required. The qubit in the right corner is initialized in $|+\rangle$. **c** Patch deformation by moving corners. The top X boundary is changed to a Z boundary by turning on Z stabilizers adjacent to previous X stabilizers. Additional $1d$ cycles are required. **d** Measuring physical qubits colored in green in X basis to shrink the patch. **e** Preparation to extend Z boundaries. Ancilla qubits colored in light gray are initialized in $|+\rangle$. **f** Expanding Z boundaries on the top and bottom. Although the weight of logical X has increased, the code distance remains $d = 3$ because the patch cannot correct error patterns that would have been corrected using a code with $d = 5$. The parity measurement of the stabilizer is repeated $1d$ times. **g** Shrink the patch by measuring physical qubits colored in green in X basis. Total surface code cycles for boundary rotation are $3d$ cycles (Color figure online)

the measurement outcomes of the newly activated stabilizers. The correction method and verification of the boundary rotation are presented in Appendix C.

The proposed method for the H_L combines the boundary rotation and transformation of subsequent measurements. The detailed implementation process is as follows. A H_L is pushed back, changing subsequent measurements. When the H_L reaches a single measurement at the end of the circuit, it is combined with the measurement by changing the basis of the measurement from $X(Z)$ to $Z(X)$. This method applies to logical operations of the universal gate set as follows. First, if a H_L is followed by a logical CNOT and patch extension, we perform a joint $X_L X_L$ or $Z_L Z_L$ measurement after the H_L. To move the H_L to the end of the circuit, we transform the subsequent joint measurement for arbitrary state $|\psi_1\rangle_L$ and $|\psi_2\rangle_L$ using the relation [14, 21, 31]

$$(H_L \otimes I_L)\,|\psi_1\rangle_L\,|\psi_2\rangle_L + (-1)^{M_{XX}}(X_L \otimes X_L)(H_L \otimes I_L)\,|\psi_1\rangle_L\,|\psi_2\rangle_L \quad (3)$$

$$(H_L \otimes I_L)\,|\psi_1\rangle_L\,|\psi_2\rangle_L + (-1)^{M_{ZZ}}(Z_L \otimes Z_L)(H_L \otimes I_L)\,|\psi_1\rangle_L\,|\psi_2\rangle_L \quad (4)$$

$$(X_L \otimes X_L)(H_L \otimes I_L) = (H_L \otimes I_L)(Z_L \otimes X_L) \quad (5)$$

$$(Z_L \otimes Z_L)(H_L \otimes I_L) = (H_L \otimes I_L)(X_L \otimes Z_L) \quad (6)$$

where M_{XX} and M_{ZZ} are the results of the joint $X_L X_L$ and $Z_L Z_L$ measurements. Therefore, H_L can be moved behind logical CNOT by transforming joint $Z_L Z_L (X_L X_L)$ to $X_L Z_L (Z_L X_L)$ measurements. A boundary rotation is required to change the type of boundaries; therefore, logical H takes $3d$ cycles. In the case of a logical CNOT after logical H in Fig. 9a, we apply the joint $X_L Z_L$ measurement with $|+\rangle_L$ instead of the joint $Z_L Z_L$ measurement. Figure 9b shows the joint $X_L Z_L$ measurement between a data patch and an ancilla patch. Second, S_L changes its joint $Z_L Y_L$

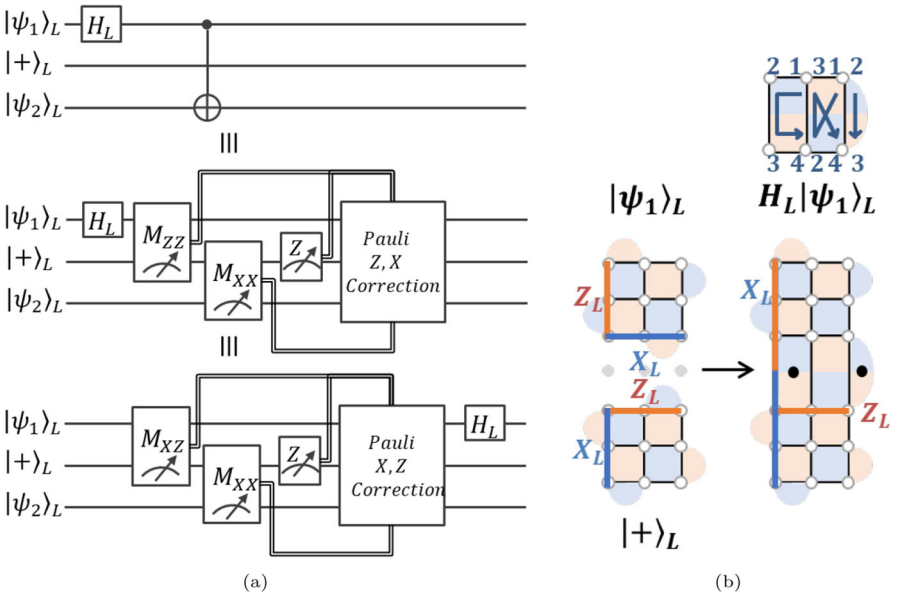


Fig. 9 **a** An example of a logical CNOT following a H_L. The H_L of $|\psi_1\rangle_L$ transforms the joint $Z_L Z_L$ measurement of the CNOT to the joint $X_L Z_L$ measurement. **b** The change of stabilizers during joint $X_L Z_L$ measurement in (**a**) and the execution order of $X_L Z_L$ stabilizers. Using the boundary rotation followed by error correction, the boundary of the patch seems to rotate 90 degrees. Merging with $|+\rangle_L$ facing $X_L X_L$ boundaries is turned into the operation facing $X_L Z_L$ boundaries

measurement to joint $X_L Y_L$ measurement to move H_L behind S_L. T_L also changes its joint $Z_L Y_L$ and $Z_L Z_L$ measurement to joint $X_L Y_L$ and $Z_L X_L$ measurement.

Figure 10 shows examples of S_L and T_L after H_L. In Fig. 10a, S_L is replaced by joint measurements and Pauli correction. After moving H_L behind S_L, a joint $Z_L Y_L$ measurement changes to a joint $X_L Y_L$ measurement, and Pauli correction is affected by the movement. The changes in joint measurements and Pauli correction for T_L are shown in Fig. 10b. To conditionally apply S_L correction, the basis of the measurement on $|0\rangle_L$ is determined depending on the result of the joint $Z_L X_L$ measurement. The joint $X_L Y_L$ measurement generated by H_L requires a boundary rotation and a 4-qubit twist defect.

Figure 11a shows stabilizers, logical operators, and a 4-qubit defect for the joint $X_L Y_L$ measurement. In Fig. 11a, the product of the parity measurement results for the stabilizers, marked with black circles, is equivalent to the $X_L Y_L$ measurement result. Z_L and X_L are defined as XXX and $ZZZXXX$, respectively. In addition, we can check the details of the 4-qubit defect in Fig. 11b. The parity measurement circuit for the defect is shown, and the order of execution is Y_1-X_2-X_3-X_4, marked by a blue arrow. Third, Pauli operations are handled in the software, which is not a consideration, and H_L, followed by another H_L, is equivalent to the logical identity operator. Finally, if the subsequent operation is a single Z or X basis measurement, H_L is absorbed by changing the basis from Z to X or from X to Z without any additional cost, as shown in Fig. 12.

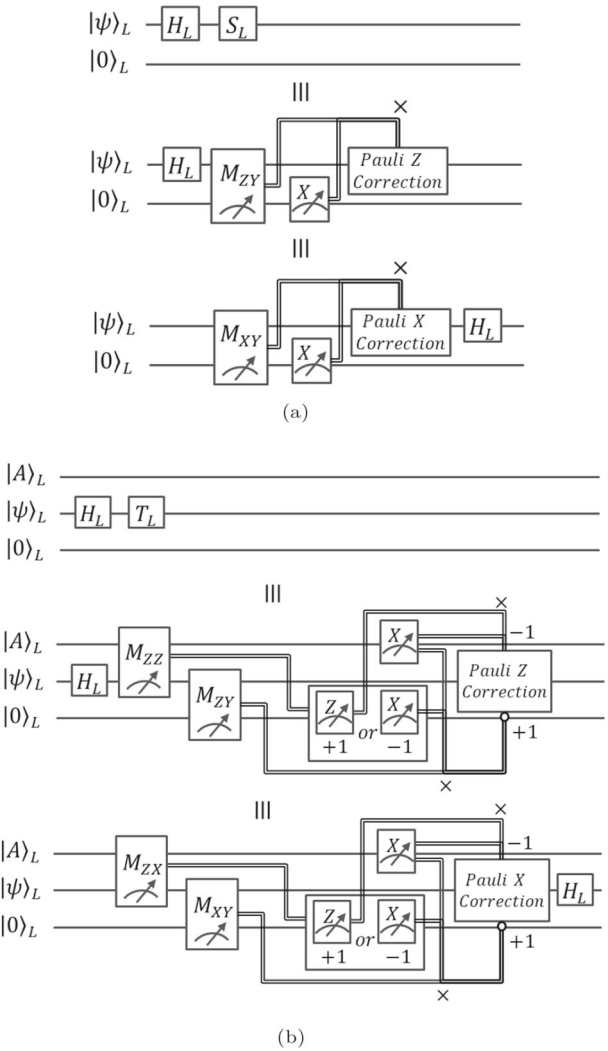


Fig. 10 **a** An example of a S_L following a H_L. The H_L of $|\psi\rangle_L$ transforms the joint $Z_L Y_L$ measurement to the joint $X_L Y_L$ measurement. The notation $\times$ indicates the product of the measurement results. In addition, Pauli correction can be changed depending on the relation with H. **b** An example of a T_L following a H_L. The H_L of $|\psi\rangle_L$ transforms the joint $Z_L Z_L$ and $Z_L Y_L$ measurement to the joint $Z_L X_L$ and $X_L Y_L$ measurement

The comparison results of implementing a H_L with previous studies are presented in Table 4. For the time cost, we calculate the surface code cycle required to complete each method, and the space cost is the number of physical qubits required per logical qubit. The first method using quantum teleportation performs H_L without physical Hadamard operations. Because logical SWAPs have been added to return to their original location after the teleportation, it takes considerable time and space costs. The second method is a more reasonable way to implement a H_L using physical

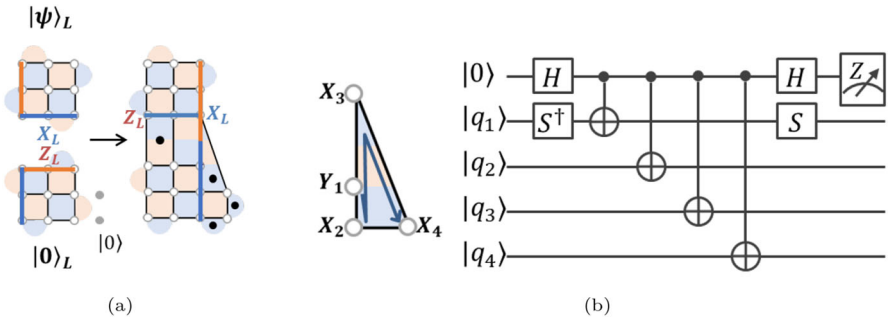


Fig. 11 **a** An example of the joint $X_L Y_L$ measurement with $|0\rangle_L$. After the boundary rotation followed by error correction, merging with $|0\rangle_L$ facing $Z_L Y_L$ boundaries is turned into the operation facing $X_L Y_L$ boundaries. **b** A 4-qubit twist defect and corresponding circuit for parity measurement. The circuit takes eight steps to finish, similar to the parity check circuit of other X and Z stabilizers [12]

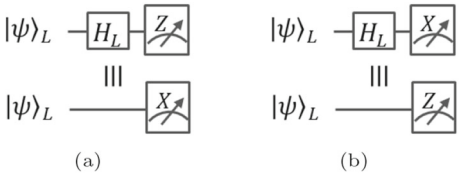


Fig. 12 **a** An example of single Z measurement following a H_L. **b** An example of single X measurement following a H_L. H_L changes the basis from X to Z or Z to X

Table 4 Comparison of the implementation of H_L operations

Method	Physical Hadamard	Time cost	Space cost (per logical qubit)	References
Quantum teleportation	×	$7d$	$2d^2$	[23]
Merging with surrounding areas	◯	$4d$	$2d^2$	[14]
Boundary rotation and transformation	×	$3d$	$2d^2 + d$	Proposed

Hadamards because it takes $4d$ cycles. As shown in Fig. 7, extra space is reserved for rotation, taking additional $1d^2$ physical qubits per patch. The authors in [16, 17] adopt this method. The proposed method combines the boundary rotation in [20] and the transformation of the subsequent measurement, requiring only $3d$ of cycles, using additional $1d^2$ physical qubits for the rotation and $1d$ physical qubits for a 4-qubit defect.

4 Numerical results

4.1 Benchmarks

The benchmarks used in the study are listed in Table 5. *4-qubit* is an arbitrarily generated 4-qubit quantum circuit, as shown in Fig. 13. *3AQFT* is a circuit for an approximate quantum Fourier transform (AQFT) using three logical qubits and

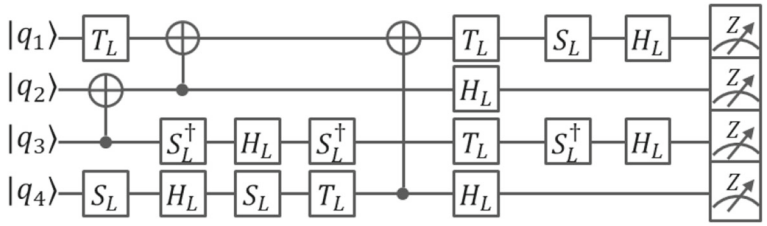


Fig. 13 An example of 4-qubit quantum circuit

Table 5 Benchmarks

Benchmark	# logical qubits	# logical gates	# S_L, T_L	# H_L	R_{tshg}%	r-arch size	t-arch size
4-qubit	4	19	10	6	84.21	$4 \times 1 \times 3$	$2 \times 2 \times 6$
3AQFT	4	36	14	14	77.78	$4 \times 1 \times 3$	$2 \times 2 \times 6$
Y-dist(7)	8	23	7	4	47.83	$8 \times 1 \times 3$	$3 \times 3 \times 6$
A-dist(15)	16	55	15	5	36.36	$16 \times 1 \times 3$	$4 \times 4 \times 6$
A-dist(20)	25	89	20	7	30.34	$25 \times 1 \times 3$	$5 \times 5 \times 6$
Shor15	6	225	105	30	60.00	$6 \times 1 \times 3$	$3 \times 3 \times 4$
Adder0-5	16	290	126	36	55.86	$16 \times 1 \times 3$	$4 \times 4 \times 6$
Adder1-8	18	273	112	32	52.75	$18 \times 1 \times 3$	$5 \times 5 \times 6$

one ancilla qubit[32, 33]. *Y-dist(7)* is a magic state Y distillation circuit that uses the [[7,1,3]] code [26]. *A-dist(15)* and *A-dist(20)* are the magic state A distillation circuits using the [[15,1,3]] code and [[20,4,2]] code, respectively [26, 28]. The remaining benchmarks are presented in [34]. Table 5 lists the architecture size of the patch units and percentage of expensive S_L, T_L, and H gates, which is expressed as $R_{tshg} = (\#S_L, T_L + \#H_L)/\#\,logical\ gates$.

4.2 Results

Table 6 summarizes the logical operations and the required time and space costs. The previous work represents the combined method of [14–17, 19, 20], which requires the lowest space-time cost. S_L and T_L consume $1d^2$ physical qubits per patch, but H_L requires physical H operations and $2d^2$ physical qubits per logical qubit. Therefore, the space cost per patch is based on $2d^2$ physical qubits per patch, and a column called physical H indicates whether physical operations are performed. Unlike the previous work, in the proposed method, $2d^2 + d$ physical qubits are enough for H_L, and $1d^2 + 2d$ physical qubits are used in S_L and T_L. Accordingly, the space cost per patch of the proposed method is $2d^2 + d$. The time cost in the second row refers to the value of previous studies, and the time costs of S_L and T_L in the third row are $1d$ cycles each because only one joint measurement is taken. H_L requires only $3d$ cycles for rotation without physical H operations, and changing the basis of subsequent measurement is handled in software, requiring no additional time cost. In both rows, the time cost of the CNOT operation uses the value presented in previous studies.

Table 6 The time and space cost of logical operations

	Time cost					Space cost	Space-time cost	Physical
	Init, Pauli, MSMT	CNOT (t-,r-arch)	H_L	S_L	T_L	(per patch)	MSD $\|Y\rangle_L$	H
Previous work	0 [14]	$4d$, $2d$[17], [16]	$4d$ [14][a)]	$1d$ [19][b)]	$2d$ [15][c)]	$2d^2$ [14]	$\geq 112d^3$ [20]	◯
Proposed method	0 [14]	$4d$, $2d$ [17], [16]	$3d$	$1d$	$1d$	$2d^2 + d$	0	×

[a)] Merging with surrounding areas, Table 4
[b)] $Z_L Z_L$ measure with $|Y\rangle_L$, Table 2
[c)] $Z_L Z_L$ measure with $|A\rangle_L$ and S_L correction, Table 3

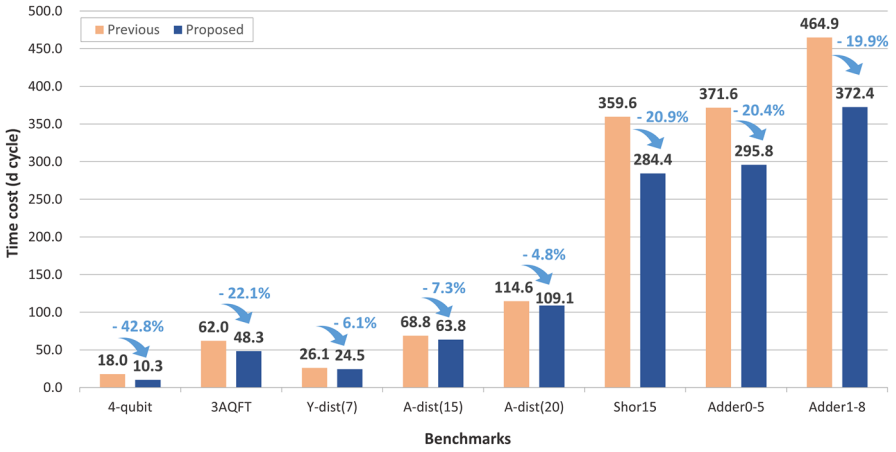


Fig. 14 The time cost of benchmarks when arranging logical qubits in r-arch

Unlike the proposed method, S_L and T_L in previous work require several magic state distillation(MSD) blocks of $|Y\rangle_L$ in addition to $|A\rangle_L$. In [20], the space-time cost of one $|Y\rangle_L$ distillation block is $28d^3$, which is the product of the time cost $4d$ and the space cost $7d^2$. In the simulation of benchmark circuits, some S_L and T_L operations can be performed simultaneously or right after prior S_L and T_L operations. To prepare for such circumstances, we must prepare at least one $|Y\rangle_L$ for each surface code cycle. Since the time cost of $|Y\rangle_L$ distillation is $4d$, four distillation blocks are required to obtain one $|Y\rangle_L$ per surface code cycle. Therefore, the minimum space-time cost of $|Y\rangle_L$ distillation blocks is $4 \times 28d^3 = 112d^3$.

The simulation is based on the t-arch and r-arch developed in [16, 17] and uses the scheduling and routing techniques presented in those studies. However, neither study considered the space for $|0\rangle_L$ used in S_L and the space for $|A\rangle_L$ used in T_L. Therefore, S_L and T_L are routed to operations using an ancilla patch. One line of ancilla patch is added to all architectures to reserve space for $|0\rangle_L$, $|Y\rangle_L$, or $|A\rangle_L$. In all simulations, the window size is fixed at 30 [16]. The time cost of the benchmarks is the average value after repeating the same circuit 10 times. The space cost of each benchmark is calculated as the product of the number of patches(architecture size) in Table 5 and the space cost per patch in Table 6. Therefore, we can calculate $space\text{-}time\ cost = (time\ cost) \times d \times (architecture\ size) \times (space\ cost\ per\ patch) + (space\text{-}time\ cost\ of\ |Y\rangle_L\ distillation)$.

Figure 14 shows the average time cost in the r-arch. Blue arrows indicate a reduction in the proposed method compared to the previous work in percentage terms, which is calculated using the formula $rate\ of\ change = 100 \times (proposed\ method - previous\ work)/previous\ work$. In the bar graph, when the circuit is operated using the proposed H_L, S_L, and T_L, the time cost is reduced by 42.8% in the randomly generated 4-qubit circuit. In other benchmarks requiring more qubits, the time cost is also reduced by at least 4.8% and by up to 22.1%. This is because the proposed method reduces the required time for H_L and T_L, and this reduction is sufficient to reduce the overall time cost, even after scheduling and routing. In Fig. 15, we verify the gain of

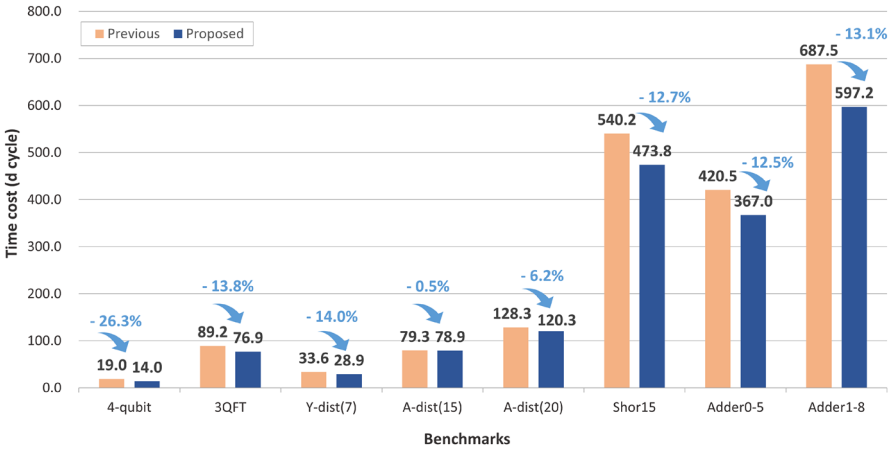
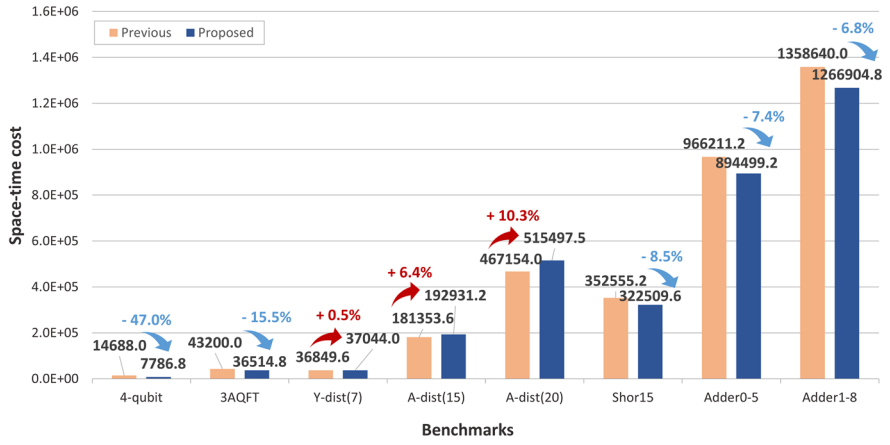


Fig. 15 The time cost of benchmarks when arranging logical qubits in t-arch

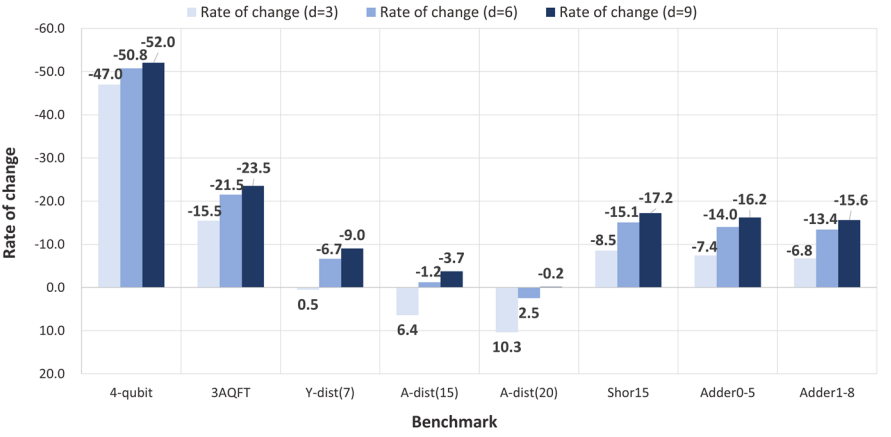
the proposed method based on the t-arch. The 4-qubit circuit consumes 26.3% lower time cost. For the other benchmarks, the time cost is reduced by a minimum of 0.5% and a maximum of 13.8%. Therefore, the proposed method provides advantages in terms of time cost for both r-arch and t-arch.

In addition, the proposed method is beneficial not only in terms of the time cost, but also in terms of the space-time cost. Figure 16a shows the space-time cost when the distance d is three. Because the architecture size is the same in the previous work and the proposed method, the number of patches required is also the same. However, the magic state distillation block for $|Y\rangle_L$ is required in the previous work. Since the space-time cost of $|Y\rangle_L$ distillation is only added to the space-time cost of the previous work, this gain is reflected in the space-time cost in all benchmarks. Therefore, the space-time cost is reduced by 47.0% in the 4-qubit circuit. In other benchmarks, the proposed method consumes a minimum of 6.8% to a maximum of 15.5% lower space-time cost than the previous work. On the other hand, for benchmarks *Y-dist(7)*, *A-dist(15)*, and *A-dist(20)*, where R_{tshg} is less than 50%, the cost does not decrease. This is because the proportion of T_L, S_L, and H_L is small, and the gain from those operations is reduced. Therefore, Fig. 16b shows the rate of changes in the space-time cost when the code distance is 3, 6, and 9. As the distance increases, the space-time cost of the proposed method is further reduced even in benchmarks where R_{tshg} is less than 50%.

When the t-arch is applied, the space-time cost is calculated using the time cost in Fig. 15 and the space cost with a distance of 3 is shown in Fig. 17a. The space-time cost is reduced by 23.4% in the 4-qubit circuit. Also, the proposed method performs other benchmarks at a lower cost from a minimum of 2.0% to a maximum of 2.7%. For *A-dist(15)*, *A-dist(20)*, *Shor15*, *Adder0-5*, and *Adder1-8*, the cost does not decrease. Since the gain in the time cost is less than the gain of r-arch and R_{tshg} is less than 60%, the benefit of the proposed method has a minimal impact on the space-time cost. However, as the distance increases from 3 to 9 in Fig. 17b, the space-time cost of the proposed method is further reduced in those benchmarks.



(a)



(b)

Fig. 16 **a** The space-time cost of r-arch using the time cost in figure 14 when the distance is 3. **b** Rate of changes in the space-time cost when the distance increases to 3, 6, and 9. More reduction means greater gain

For both r-arch and t-arch, the time cost is reduced in all benchmarks, and the benefit of the space-time cost is enlarged as the distance increases. In addition to the benefit of the cost, the proposed method does not require physical H operations when implementing logical H, so errors due to physical H operations can be ignored.

5 Conclusion

In this study, we propose a method for performing S_L, T_L, and H_L at a low cost. Logical operations are decomposed into joint measurements and implemented at the

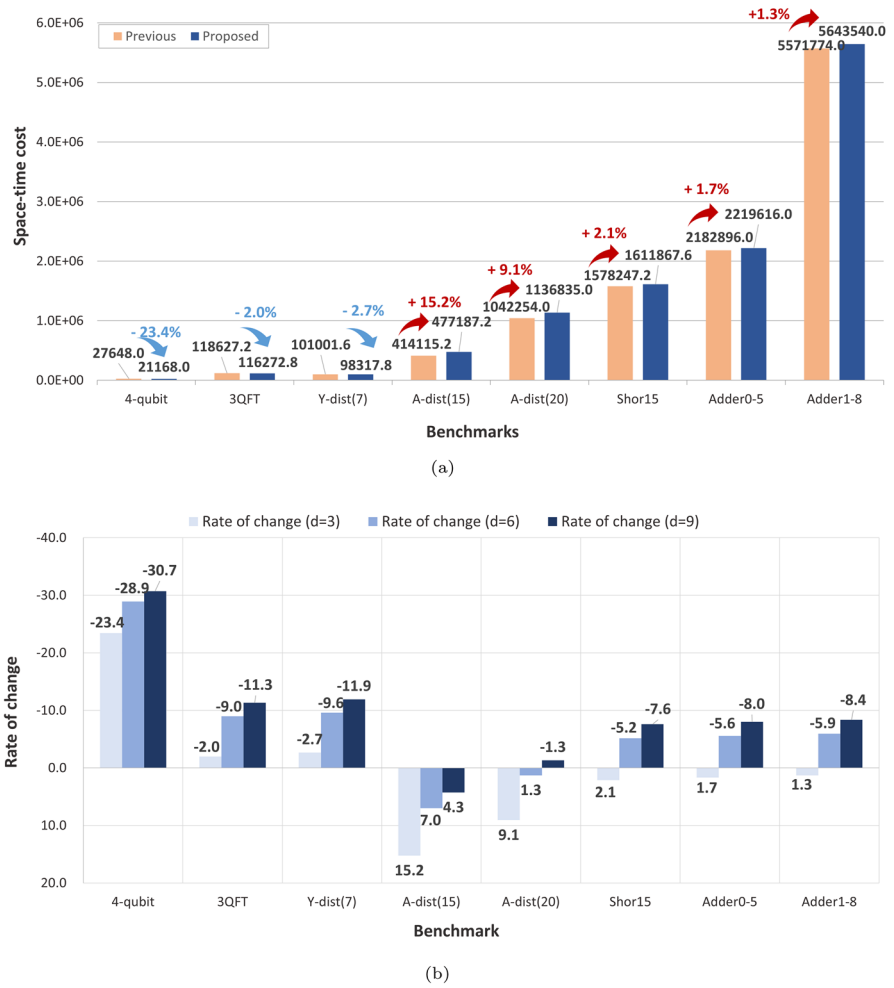


Fig. 17 **a** The space-time cost of t-arch using the time cost in Fig. 15 when the distance is 3. **b** Rate of changes in the space-time cost when the distance increases to 3, 6, and 9. More reduction means greater gain

lattice surgery level. In other words, we describe the newly activated stabilizers during merging, specify the correction method for the X_L and Z_L operators according to the result of the parity measurement, and indicate the physical qubits to be measured. In addition, a boundary rotation that changes the stabilizers of boundaries is introduced, resulting in implementing a H_L with minimal time cost. The proposed method is described based on a rotated surface code with a distance of 3, which can be extended to a higher distance, and examples are presented in Appendix D.

In terms of time cost, S_L and T_L can be executed at a low cost of $1d$ cycles, and H_L can be executed in $3d$ cycles, which is less than in the previous $4d$ cycles. When the proposed method is applied to various benchmarks, the time cost is reduced even

if they go through scheduling and routing to find operations that can be performed simultaneously. In the r-arch, we can perform the 4-qubit circuit in 42.8% less time, whereas other benchmarks can be performed in up to 22.1% less time. In the t-arch, benchmarks can be performed in 26.3% less time for the 4-qubit circuit, and other circuits can be performed in up to 13.8% less time. Regarding the space cost, the number of patches is fixed according to the type of architecture; therefore, there is a difference in the space and time cost for $|Y\rangle_L$ distillation blocks. In the previous work, because $2d^2$ physical qubits are required to implement a H_L, all patches must reserve at least $2d^2$ physical qubits. However, $2d^2 + d$ physical qubits are sufficient to implement a H_L in the proposed method, and $1d^2 + 2d$ physical qubits are required for S_L and T_L. As a result, the space-time cost of the 4-qubit circuit is reduced by 47.0% in the r-arch and 23.4% in the t-arch. Additionally, the space-time cost of other benchmarks is reduced by up to 23.5% in the r-arch and up to 11.9% in the t-arch, as the code distance increases to 9.

In future work, beyond the joint measurement between two patches, we can apply boundary rotation and $Z_L Y_L$ measurement to a multi-patch measurement performed simultaneously across multiple patches. Therefore, they can be applied not only to the r-arch and t-arch, but also to the architecture for multi-patch measurement [20] and column-type architecture for multi-target CNOT [26]. Although this study is based on the rotated surface code, it is expected to be expanded in lattice surgery of the triangular code [35] or the color code [36].

Appendix A 3D space-time figure of logical operators during the joint $Z_L Y_L$ measurement

First, after joint $Z_L Y_L$ measurement, we can find equivalent forms of logical operators by multiplying them by the product of the stabilizer. Figure 18 shows the logical X, which is equal to the original logical X multiplied by the product of the stabilizers marked in blue dots. Figure 19 shows the equivalent logical Z multiplied by the product of the stabilizer marked in orange dots. In Fig. 20, we present a 3D space-time figure where the logical operators in Figs. 18 and 19e, f are supported. The dark blue wall indicates where the logical X operator is supported, and the light orange wall indicates where the logical Z operator is supported. Figure 20 shows that the distance between the logical X and Z operators is maintained at d.

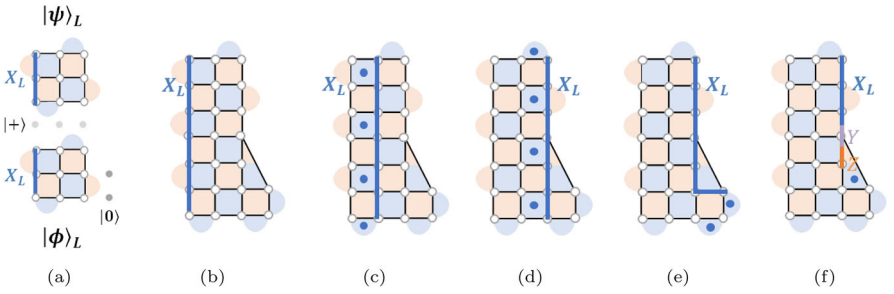


Fig. 18 Variations of logical X, X_L, by the product of the stabilizers during the joint $Z_L Y_L$ measurement **a** $X_L = XXX$ before the joint $Z_L Y_L$ measurement, **b** $X_L = XXXXXXX$ after the joint $Z_L Y_L$ measurement, **c**–**e** variations of logical X by the product of the stabilizers marked in blue, **f** variation of logical X by the product of the stabilizers marked in blue, where $X_L = XXXYZ$ (Color figure online)

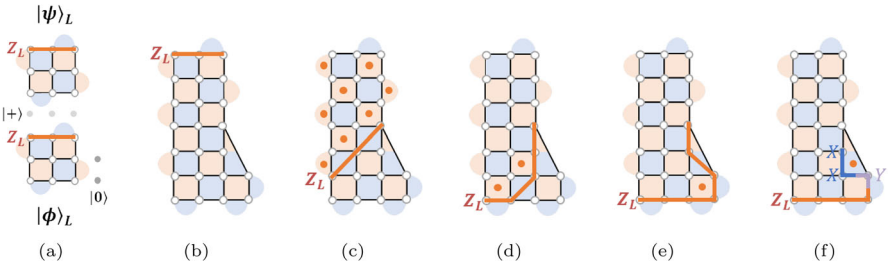


Fig. 19 Variations of logical Z, Z_L, by the product of the stabilizers during the joint $Z_L Y_L$ measurement **a** $Z_L = ZZZ$ before the joint $Z_L Y_L$ measurement, **b** $Z_L = ZZZ$ after the joint $Z_L Y_L$ measurement, **c**–**e** variations of logical Z by the product of the stabilizers marked in orange, **f** variation of logical Z by the product of the stabilizers marked in orange, where $Z_L = XXYZZZZ$ (Color figure online)

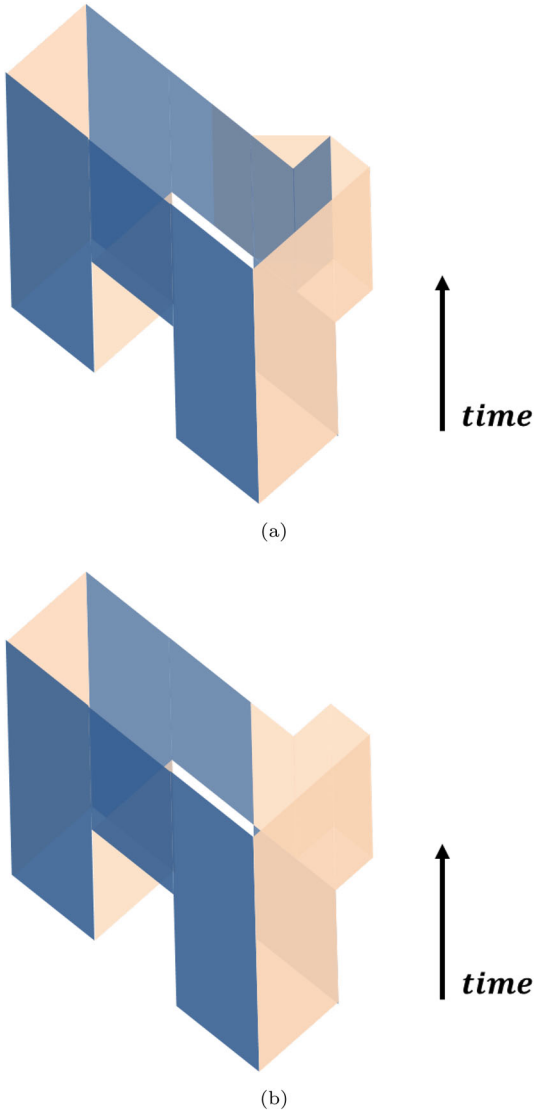


Fig. 20 3D space-time figure for the joint $Z_L Y_L$ measurement. Dark blue wall represents where logical X is supported; light orange wall represents where logical Z is supported. **a** 3D space-time figure using the logical operators in Figs. 18e and 19e. **b** 3D space-time figure using the logical operators in Figs. 18f and 19f (Color figure online)

Appendix B Stabilizer description of the S_L for a distance-3 code

Logical S can be proven by satisfying $S_L X_L S_L^\dagger = Y_L$ and $S_L Z_L S_L^\dagger = Z_L$. Table 7 lists the stabilizers for one data patch in an arbitrary state and one ancilla patch in $|0\rangle_L$. If we track the changes in X_{L1} and Z_{L1}, which are the logical X and Z of the data patch, respectively, we can determine whether logical S has been performed. In the figure on the left, a,b, and c are the ancilla qubits prepared in $|+\rangle_L$, and $K_a \sim K_c$ are the corresponding stabilizers. d and e are ancilla qubits prepared in $|0\rangle_L$ and $K_d \sim K_e$ are the corresponding stabilizers.

Table 7 Stabilizer description of one data patch and one ancilla patch in $|0\rangle_L$

	1	2	3	4	5	6	7	8	9	a	b	c	1	2	3	4	5	6	7	8	9	d	e
K_1	X	X		X	X																		
K_2				X	X		X	X															
K_3		X	X																				
K_4						X	X																
K_5		Z	Z		Z	Z																	
K_6				Z	Z		Z	Z															
K_7	Z			Z																			
K_8						Z			Z														
X_{L1}			X		X				X														
Z_{L1}							Z	Z	Z														
K_a										X													
K_b											X												
K_c												X											
K_9													X	X		X	X						
K_{10}																	X	X		X	X		
K_{11}														X	X								
K_{12}																				X	X		
K_{13}														Z	Z		Z	Z					
K_{14}																Z	Z		Z	Z			
K_{15}													Z			Z							
K_{16}																		Z			Z		
K_{17}													Z	Z	Z								
K_d																						Z	
K_e																							Z

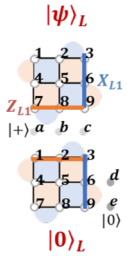


Table 8 shows X and Z stabilizers in the process of performing a joint $Z_L Y_L$ measurement. The changed parts compared to Table 7 are marked in red, and $K_c X_{L1}$ is equal to X_{L1} multiplied by the stabilizer.

Table 9 describes all stabilizers required for the joint $Z_L Y_L$ measurement, including a 4-qubit defect. While Z_{L1} remains unchanged, X_{L1} is transformed into the chain shown in the figure on the left. When X_{L1} is further multiplied by stabilizers $K_{21} \sim K_{23}$, it is transformed into $K_{21}K_{22}K_{23}K_{17}K_c X_{L1}$ in Table 10.

If we multiply this by $K_{18} \sim K_{20}$, it transforms into Y_L in Table 11. Thus, X_{L1} is replaced by $Y_L = K_{18}K_{19}K_{20}K_{21}K_{22}K_{23}K_{17}K_c X_{L1}$ of the unified patch, and Z_{L1} becomes Z_L, confirming that S_L is performed. In addition, logical operators are calibrated according to the results of the parity measurements of new stabilizers, and Pauli correction is performed on the entire patch according to the measurement outcomes of the physical qubits in Fig. 5c.

Table 8 Stabilizer description with new X and Z stabilizers during the joint $Z_L Y_L$ measurement

		1	2	3	4	5	6	7	8	9	a	b	c	1	2	3	4	5	6	7	8	9	d	e
K_1		X	X		X	X																		
K_2						X	X		X	X														
K_3			X	X																				
$K_a K_b K_4$								X	X		X	X												
K_5			Z	Z		Z	Z																	
K_6					Z	Z		Z	Z															
K_7		Z			Z																			
K_8							Z			Z														
$K_c X_{L1}$				X			X			X			X											
Z_{L1}								Z	Z	Z														
K_{18}	$(-1)^{m_{18}}$							Z			Z													
K_{19}	$(-1)^{m_{19}}$								Z	Z		Z	Z											
K_{20}	$(-1)^{m_{20}}$										Z	Z		Z	Z									
K_9														X	X		X	X						
K_{10}																		X	X		X	X		
$K_b K_c K_{11}$												X	X		X	X								
K_{12}																				X	X			
K_{13}															Z	Z		Z	Z					
K_{14}																	Z	Z		Z	Z			
K_{15}														Z			Z							
$K_d K_e K_{16}$																			Z			Z	Z	Z
K_{17}														Z	Z	Z								
K_{21}	$(-1)^{m_{21}}$																						X	X
K_{22}	$(-1)^{m_{22}}$																					X		X

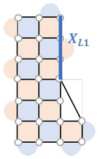


Table 9 Stabilizer description with 4-qubit defect after joint $Z_L Y_L$ measurement

		1	2	3	4	5	6	7	8	9	a	b	c	1	2	3	4	5	6	7	8	9	d	e
K_1		X	X		X	X																		
K_2						X	X		X	X														
K_3			X	X																				
$K_a K_b K_4$								X	X		X	X												
K_5			Z	Z		Z	Z																	
K_6					Z	Z		Z	Z															
K_7		Z			Z																			
K_8							Z			Z														
$K_{17} K_c X_{L1}$				X			X			X			X	Z	Z	Z								
Z_{L1}								Z	Z	Z														
K_{18}	$(-1)^{m_{18}}$							Z			Z													
K_{19}	$(-1)^{m_{19}}$								Z	Z		Z	Z											
K_{20}	$(-1)^{m_{20}}$										Z	Z		Z	Z									
K_9														X	X		X	X						
K_{10}																		X	X		X	X		
$K_b K_c K_{11}$												X	X		X	X								
K_{12}																				X	X			
K_{13}															Z	Z		Z	Z					
K_{14}																	Z	Z		Z	Z			
K_{15}														Z			Z							
$K_d K_e K_{16}$																			Z			Z	Z	Z
K_{21}	$(-1)^{m_{21}}$																						X	X
K_{22}	$(-1)^{m_{22}}$																					X		X
K_{23}	$(-1)^{m_{23}} i$												Z			XZ			X				X	

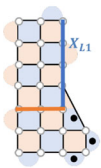


Table 10 Stabilizer description of a unified patch after joint $Z_L Y_L$ measurement

		1	2	3	4	5	6	7	8	9	a	b	c	1	2	3	4	5	6	7	8	9	d	e
K_1		X	X		X	X																		
K_2						X	X		X	X														
K_3			X	X																				
$K_a K_b K_4$								X	X		X	X												
K_5			Z	Z		Z	Z																	
K_6					Z	Z		Z	Z															
K_7		Z			Z																			
K_8							Z			Z														
$K_{21}K_{22}K_{23}$ $K_{17}K_c X_{L1}$	$(-1)^{\binom{m_{21}+m_{22}}{+m_{23}}} i$			X			X			X			ZX	Z	Z	X			X			X		
Z_{L1}								Z	Z	Z														
K_{18}	$(-1)^{m_{18}}$							Z			Z													
K_{19}	$(-1)^{m_{19}}$								Z	Z		Z	Z											
K_{20}	$(-1)^{m_{20}}$										Z	Z		Z	Z									
K_9														X	X		X	X						
K_{10}																		X	X		X	X		
$K_b K_c K_{11}$												X	X		X	X								
K_{12}																				X	X			
K_{13}															Z	Z		Z	Z					
K_{14}																	Z	Z		Z	Z			
K_{15}														Z			Z							
$K_d K_e K_{16}$																			Z			Z	Z	Z
K_{21}	$(-1)^{m_{21}}$																						X	X
K_{22}	$(-1)^{m_{22}}$																					X		X
K_{23}	$(-1)^{m_{23}} i$												Z			XZ			X				X	

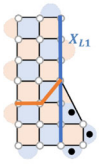
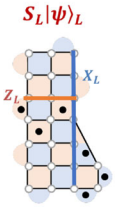


Table 11 Stabilizer description of a unified patch which is equal to Table 10 multiplied by stabilizers

		1	2	3	4	5	6	7	8	9	a	b	c	1	2	3	4	5	6	7	8	9	d	e
K_1		X	X		X	X																		
K_2						X	X		X	X														
K_3			X	X																				
$K_aK_bK_4$								X	X		X	X												
K_5			Z	Z		Z	Z																	
K_6					Z	Z		Z	Z															
K_7		Z			Z																			
K_8							Z			Z														
Y_L	$(-1)^{\binom{m_{18}+m_{19}+}{m_{20}+m_{21}+}{m_{22}+m_{23}+1}i}$			X			X	Z	Z	XZ			X			X			X			X		
Z_L								Z	Z	Z														
K_{18}	$(-1)^{m_{18}}$							Z			Z													
K_{19}	$(-1)^{m_{19}}$								Z	Z		Z	Z											
K_{20}	$(-1)^{m_{20}}$										Z	Z		Z	Z									
K_9														X	X		X	X						
K_{10}																		X	X		X	X		
$K_bK_cK_{11}$												X	X		X	X								
K_{12}																				X	X			
K_{13}															Z	Z		Z	Z					
K_{14}																	Z	Z		Z	Z			
K_{15}														Z			Z							
$K_dK_eK_{16}$																			Z			Z	Z	Z
K_{21}	$(-1)^{m_{21}}$																						X	X
K_{22}	$(-1)^{m_{22}}$																					X		X
K_{23}	$(-1)^{m_{23}}i$												Z			XZ			X				X	



Appendix C The correction and verification for the boundary rotation

During the boundary rotation, we must examine error correction, logical operators, and code distance [37, 38]. The code distance of the original information in Fig. 21 is 3, and logical operators $X_{L_0} = X_1X_4X_7$ and $Z_{L_0} = Z_7Z_8Z_9$. Since ancilla qubits 10 $\sim$ 11, 13 $\sim$ 18 are initialized in $|0\rangle$, which is $+1$ eigenstate of single Z stabilizers, Z errors cannot damage those qubits.

Figure 22 shows logical operators and stabilizers after extending the X boundary on the top. For example, an X stabilizer on qubits 3, 6, 10, and 13 is expressed as $K_1 = X_3X_6X_{10}X_{13}$ and the parity of K_1 is $(-1)^{m_1}$. The logical operators Z_{L_1} and X_{L_1} are equivalent to Z_{L_0} and X_{L_0}. X stabilizers marked in diagonal lines($K_1 = X_3X_6X_{10}X_{13}$, $K_2 = X_{11}X_{12}$, $K_3 = X_{11}X_{14}X_{15}$, $K_4 = X_{13}X_{14}X_{16}X_{17}$, and $K_5 = X_{15}X_{18}$) can have $+1$ or -1 parity without Z errors. Therefore, we can detect Z errors from the following surface code cycle when the sign of those X stabilizers is inverted.

Figure 23 shows the process of transforming the stabilizers of the top boundary. Z and X errors can be detected by existing stabilizers. However, we cannot distinguish whether the parities of $K_6 = Z_1Z_2$, $K_7 = Z_3Z_{10}$ and $K_8 = Z_{11}Z_{12}$ are -1

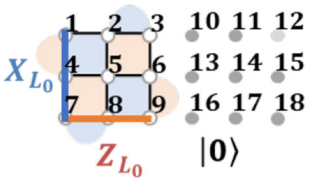


Fig. 21 A logical qubit in arbitrary states encoded by the surface code with a distance of 3. The numbers correspond to each qubit. The logical operators are expressed as $X_{L_0} = X_1X_4X_7$ and $Z_{L_0} = Z_7Z_8Z_9$

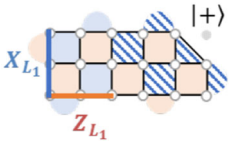


Fig. 22 Extending the top X boundary. The measurement outcome of stabilizers with diagonal lines can be $+1$ or -1. Logical operators are expressed as $Z_{L_1} = Z_{L_0}$ and $X_{L_1} = X_{L_0}$. Z errors do not affect ancilla qubits and X errors can be detected by Z stabilizers

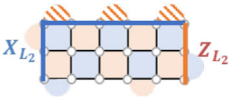


Fig. 23 Transforming the stabilizers of the top boundary. The measurement outcome of Z stabilizers with diagonal lines can be $+1$ or -1. Gauge-fixing operators are $X_2X_3X_{10}X_{11}X_{12}$, $X_{10}X_{11}X_{12}$, and X_{12}. We can use logical operators Z_{L_2} and X_{L_2} multiplied by Z and X stabilizers

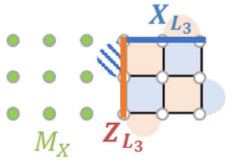


Fig. 24 Measuring qubit $1 \sim 9$ in X basis. The parity of $K_9 = X_{10}X_{13}$ can be extracted from $K_{10} = X_3$, $K_{11} = X_6$, and $K_1 = X_3X_6X_{10}X_{13}$. The logical operators Z_{L_3} and X_{L_3} are equivalent to Z_{L_2} and X_{L_2} by the product of Z or X stabilizers

due to an X error, or whether the state is projected into -1 eigenspace without X errors. Therefore, the patch is fixed using $X_2X_3X_{10}X_{11}X_{12}$, $X_{10}X_{11}X_{12}$, and X_{12}. The logical operator Z_{L_2} is the product of Z_{L_1} and Z stabilizers including those marked in diagonal lines and expressed as $Z_{L_2} = (-1)^{m_6+m_7+m_8}Z_{12}Z_{15}Z_{18}$. The logical operator X_{L_2} is also the product of X_{L_1} and X stabilizers and expressed as $X_{L_2} = (-1)^{m_2}X_1X_2X_3X_4X_7X_{10}X_{11}X_{12}$, which is handled in software [15]. The weight of the minimum logical operator is 3, and some error patterns in which the weight is greater than one cannot be accurately distinguished.

In Fig. 24, qubits $1 \sim 9$ are measured in X basis. X errors have no effect on those measurements, and Z errors are detected by the inverted sign of measurement outcomes. Using X measurement results of qubit 3 and qubit 6 as well as $K_1 = X_3X_6X_{10}X_{13}$, the parity of $K_9 = X_{10}X_{13}$ can be determined. If the par-

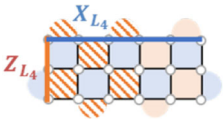


Fig. 25 Extending the patch using ancilla qubits prepared in $|+\rangle$. The logical operator X_{L_4} is extended by stretching Z boundaries. Z_{L_4} is equivalent to Z_{L_3} by the product of Z stabilizers, including those marked in diagonal lines

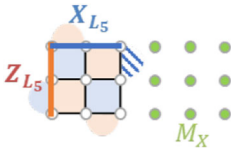


Fig. 26 Measuring qubit 10 ∼ 18 in X basis. The parity of $K_{19} = X_3X_6$ is derived from the measurement results of qubit 10 and 13, and the gauge-fixing operator is Z_3Z_{10}

ity is -1, we correct the logical patch using the gauge-fixing operator Z_3Z_{10}. The logical operator Z_{L_3} is expressed as $(-1)^{m_6+m_7}Z_{10}Z_{13}Z_{16}$ using the product of Z stabilizers. X_{L_3} is multiplied by the product of X stabilizers and expressed as $X_{L_3} = (-1)^{m_2+m_{10}+m_{11}+m_{12}}X_{10}X_{11}X_{12}$, where $K_{12} = X_9$.

Ancilla qubits are prepared in $|+\rangle$, which is $+1$ eigenstate of the single X stabilizer. X errors do not affect ancilla qubits, and Z errors can be detected by X stabilizers. The patch is extended by stretching Z boundaries, as shown in Fig. 25. The logical operator X_{L_4} is also extended as $(-1)^{m_2+m_{10}+m_{11}+m_{12}}X_1X_2X_3X_{10}X_{11}X_{12}$. The newly activated Z stabilizers marked in diagonal lines, $K_{13} = Z_1Z_2$, $K_{14} = Z_2Z_3Z_5Z_6$, $K_{15} = Z_3Z_{10}$, $K_{16} = Z_4Z_5Z_7Z_8$, $K_{17} = Z_8Z_9$, and $K_{18} = Z_6Z_9Z_13Z_16$, can have a $+1$ or -1 parity without X errors. Therefore, Z_{L_4} is equivalent to $(-1)^{m_6+m_7+m_{13}+m_{14}+m_{15}+m_{16}+m_{17}+m_{18}}Z_1Z_4Z_7$. The minimum weight of logical operators is still three.

In Fig. 26, qubit 10 ∼ 18 are measured in X basis, and the logical qubit is fixed using Z_3Z_{10} when the parity of $K_{19} = X_3X_6$ derived from the measurement results of qubit 10 and 13 is -1. The logical operator Z_{L_5} is equivalent to Z_{L_4}, and X_{L_5} is corrected as $(-1)^{m_2+m_{10}+m_{11}+m_{12}+m_{20}+m_{21}+m_{22}}X_1X_2X_3$, where $K_{20} = X_{10}$, $K_{21} = X_{11}$, and $K_{22} = X_{12}$. The process of returning from Figs. 21, 22, 23, 24, 25, and 26 proceeds similarly, starting by increasing the bottom Z boundary.

Appendix D Joint X_LZ_L and Z_LY_L measurements for a surface code with a distance larger than three

In Sect. 3, the proposed method is described based on SC-17, whose distance is 3, but the same techniques can be applied to logical qubits encoded by the surface code with a distance of three or higher. For example, logical qubits with a distance of 6 are shown in Fig. 27. Figure 27b, c shows examples of the X_LZ_L and Z_LY_L measurements, respectively.

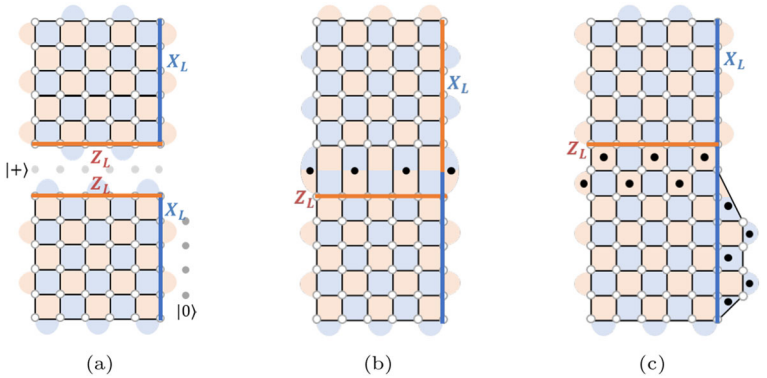


Fig. 27 **a** Two logical qubits in arbitrary states encoded by the surface code with a distance of 6. **b** A $X_L Z_L$ measurement between two logical qubits. The boundary rotation is applied to the upper logical qubit. The X_L of the unified patch is defined as the product of six Zs and six Xs, which are marked in orange and blue lines. The product of stabilizers marked in black circles is equivalent to the outcome of the $X_L Z_L$ measurement. **c** A $Z_L Y_L$ measurement between two logical qubits. The stabilizer in the bottom right-hand corner consists of three qubits, unlike the case where the distance is 3. Therefore, a 3-qubit stabilizer is used at the bottom corner of the column with a 4-qubit defect when the distance is an even number. The product of stabilizers marked in black circles is also equivalent to the outcome of the $Z_L Y_L$ measurement (Color figure online)

Acknowledgements This work was supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) [No. 2020-0-00014, A Technology Development of Quantum OS for Fault-tolerant Logical Qubit Computing Environment]. This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2024-2021-0-01810) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation). This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2023-00242396).

Author Contributions Kang, Lee, and Ha developed the scheme presented in this paper and prepared all figures and tables. Kang wrote the main manuscript text and performed a simulation to show the efficiency of the proposed scheme. Heo verified the proofs and simulation results and managed all communication between the Journal and all co-authors. All authors reviewed the manuscript.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

1. Shor, P.W.: Fault-tolerant quantum computation. In: Proceedings of 37th Conference on Foundations of Computer Science, pp. 56–65 (1996). IEEE
2. Preskill, J.: Reliable quantum computers. Proc. R. Soc. Lond. Ser. A: Math. Phys. Eng. Sci. **454**(1969), 385–410 (1998)
3. Gottesman, D.: Opportunities and challenges in fault-tolerant quantum computation. arXiv preprint arXiv:2210.15844 (2022)
4. Wang, D.S., Fowler, A.G., Hollenberg, L.C.: Surface code quantum computing with error rates over 1%. Phys. Rev. A **83**(2), 020302 (2011)
5. Bourassa, J.E., Alexander, R.N., Vasmer, M., Patil, A., Tzitrin, I., Matsuura, T., Su, D., Baragiola, B.Q., Guha, S., Dauphinais, G.: Blueprint for a scalable photonic fault-tolerant quantum computer. Quantum **5**, 392 (2021)
6. Kitaev, A.Y.: Fault-tolerant quantum computation by anyons. Ann. Phys. **303**(1), 2–30 (2003)
7. Raussendorf, R., Harrington, J.: Fault-tolerant quantum computation with high threshold in two dimensions. Phys. Rev. Lett. **98**(19), 190504 (2007)
8. Krinner, S., Lacroix, N., Remm, A., Di Paolo, A., Genois, E., Leroux, C., Hellings, C., Lazar, S., Swiadek, F., Herrmann, J.: Realizing repeated quantum error correction in a distance-three surface code. Nature **605**(7911), 669–674 (2022)
9. Bravyi, S., Englbrecht, M., König, R., Peard, N.: Correcting coherent errors with surface codes. npj Quantum Inf. **4**(1), 55 (2018)
10. Tomita, Y., Svore, K.M.: Low-distance surface codes under realistic quantum noise. Phys. Rev. A **90**(6), 062320 (2014)
11. Nielsen, M.A., Chuang, I.: Quantum computation and quantum information. American Association of Physics Teachers (2002)
12. Fowler, A.G., Mariantoni, M., Martinis, J.M., Cleland, A.N.: Surface codes: towards practical large-scale quantum computation. Phys. Rev. A **86**(3), 032324 (2012)
13. Litinski, D., Oppen, F.: Lattice surgery with a twist: simplifying Clifford gates of surface codes. Quantum **2**, 62 (2018)
14. Horsman, C., Fowler, A.G., Devitt, S., Van Meter, R.: Surface code quantum computing by lattice surgery. New J. Phys. **14**(12), 123011 (2012)
15. Fowler, A.G., Gidney, C.: Low overhead quantum computation using lattice surgery. arXiv preprint arXiv:1808.06709 (2018)
16. Lee, J., Kang, Y., Ha, J., Heo, J.: Lattice surgery-based surface code architecture using remote logical cnot operation. Quantum Inf. Process. **21**(6), 217 (2022)
17. Lao, L., Wee, B., Ashraf, I., Someren, J., Khammassi, N., Bertels, K., Almudever, C.G.: Mapping of lattice surgery-based quantum circuits on surface code architectures. Quantum Sci. Technol. **4**(1), 015005 (2018)
18. Bravyi, S., Kitaev, A.: Universal quantum computation with ideal Clifford gates and noisy ancillas. Phys. Rev. A **71**(2), 022316 (2005)
19. Haah, J., Hastings, M.B.: Measurement sequences for magic state distillation. Quantum **5**, 383 (2021)
20. Litinski, D.: A game of surface codes: large-scale quantum computing with lattice surgery. Quantum **3**, 128 (2019)
21. Zhou, X., Leung, D.W., Chuang, I.L.: Methodology for quantum logic gate construction. Phys. Rev. A **62**(5), 052316 (2000)
22. Beverland, M.E., Huang, S., Kliuchnikov, V.: Fault tolerance of stabilizer channels. arXiv preprint arXiv:2401.12017 (2024)
23. Erhard, A., Poulsen Nautrup, H., Meth, M., Postler, L., Stricker, R., Stadler, M., Negnevitsky, V., Ringbauer, M., Schindler, P., Briegel, H.J.: Entangling logical qubits with lattice surgery. Nature **589**(7841), 220–224 (2021)
24. Riesebos, L., Fu, X., Varsamopoulos, S., Almudever, C.G., Bertels, K.: Pauli frames for quantum computer architectures. In: Proceedings of the 54th Annual Design Automation Conference 2017, pp. 1–6 (2017)
25. Chamberland, C., Campbell, E.T.: Universal quantum computing with twist-free and temporally encoded lattice surgery. PRX Quantum **3**(1), 010331 (2022)
26. Herr, D., Nori, F., Devitt, S.J.: Lattice surgery translation for quantum computation. New J. Phys. **19**(1), 013034 (2017)

27. Trout, C.J., Brown, K.R.: Magic state distillation and gate compilation in quantum algorithms for quantum chemistry. Int. J. Quantum Chem. **115**(19), 1296–1304 (2015)
28. Bravyi, S., Haah, J.: Magic-state distillation with low overhead. Phys. Rev. A **86**(5), 052329 (2012)
29. Haah, J., Hastings, M.B.: Codes and protocols for distilling t, controlled-s, and Toffoli gates. Quantum **2**, 71 (2018)
30. Bravyi, S., Smith, G., Smolin, J.A.: Trading classical and quantum computational resources. Phys. Rev. X **6**(2), 021043 (2016)
31. Adedoyin, A., Ambrosiano, J., Anisimov, P., Casper, W., Chennupati, G., Coffrin, C., Djidjev, H., Gunter, D., Karra, S., Lemons, N., et al.: Quantum algorithm implementations for beginners. arXiv preprint arXiv:1804.03719 (2018)
32. Nam, Y., Su, Y., Maslov, D.: Approximate quantum Fourier transform with o (n log (n)) t gates. NPJ Quantum Inf. **6**(1), 26 (2020)
33. García-Martín, D., Sierra, G.: Five experimental tests on the 5-qubit ibm quantum computer. arXiv preprint arXiv:1712.05642 (2017)
34. Lin, C.-C., Chakrabarti, A., Jha, N.K.: Qlib: quantum module library. ACM J. Emerg. Technol. Comput. Syst. **11**(1), 1–20 (2014)
35. Yoder, T.J., Kim, I.H.: The surface code with a twist. Quantum **1**, 2 (2017)
36. Landahl, A.J., Ryan-Anderson, C.: Quantum computing by color-code lattice surgery. arXiv preprint arXiv:1407.5103 (2014)
37. Vuillot, C., Lao, L., Criger, B., Almudéver, C.G., Bertels, K., Terhal, B.M.: Code deformation and lattice surgery are gauge fixing. New J. Phys. **21**(3), 033028 (2019)
38. Gupta, R.S., Sundaresan, N., Alexander, T., Wood, C.J., Merkel, S.T., Healy, M.B., Hillenbrand, M., Jochym-O'Connor, T., Wootton, J.R., Yoder, T.J.: Encoding a magic state with beyond break-even fidelity. Nature **625**(7994), 259–263 (2024)