Received: 21 March 2024 | Revised: 20 June 2024 | Accepted: 19 August 2024

DOI: 10.4218/etrij.2024-0129

FEATURED ARTICLE

ETRI Journal WILEY

Optimal execution of logical Hadamard with low-space overhead in rotated surface code

Sang-Min Lee | **Ki-Sung Jin** | **Soo-Cheol Oh** | **Jin-Ho On** | **Gyu-Il Cha**

Future Computing Research Division, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea

Correspondence
Sang-Min Lee, Future Computing Research Division, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea.
Email: sanglee@etri.re.kr

Funding information
This work was supported by Institute of 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).

Abstract

Fault-tolerant quantum computation requires error-correcting codes that enable reliable universal quantum operations. This study introduces a novel approach that executes the logical Hadamard with low-space requirements while preserving the original definition of logical operators within the framework of the rotated surface codes. Our method leverages a boundary deformation method to rotate the logical qubit transformed by transversal Hadamard. Following this, the original encoding of the logical qubit is reinstated through logical flip-and-shift operations. The estimated space–time cost for a logical Hadamard operation with a code distance d is $5d^2 + 3d^2$. The efficiency enhancement of the proposed method is approximately four times greater than those of previous approaches, regardless of the code distance. Unlike the traditional method, implementing a logical Hadamard requires only two patches instead of seven. Furthermore, the proposed method ensures the parallelism of quantum circuits by preventing interferences between adjacent logical data qubits.

KEYWORDS

lattice surgery, logical Hadamard, quantum circuit, quantum error correction code, surface code

1 | INTRODUCTION

Quantum error correction (QEC) is a fault-tolerant quantum computing technique [1–9] developed primarily to protect quantum information from errors induced by quantum noise and fast decoherence. QEC codes achieve this by encoding multiple physical qubits into a single logical qubit, thereby enabling the detection and correcting of errors during the execution of quantum algorithms. Among the various QEC codes, surface codes [10–18] are regarded as the most promising due to their high error thresholds. The surface code generates a logical qubit using the interactions between neighboring physical qubits in a two-dimensional array. The data qubits encode the logical state, whereas the ancilla qubits periodically track the error syndrome measurements of neighboring data qubits. These syndromes can be categorized as bit-flip errors (X syndromes) or phase-flip errors (Z syndromes).

Surface codes offer a set of logical operations for universal fault-tolerant quantum computations, including logical Clifford gates (X, Z, *Hadamard*, S, and Controlled-NOT (CNOT)) and logical non-Clifford T-gates. Our research focuses on finding effective methodologies to support logical Hadamard operations (logical-H). Until recently, various studies have reported logical operation methods for logical-H, including transversal [19, 20], code defects [14–18, 21], gate teleportation [22, 23], and code deformation [24–29]. Our study focuses on logical-*H* techniques based on lattice surgery.

The transversal approach is a simple technique that enables the implementation of logical-*H*s in the context of surface codes. The transversal method achieves logical-*H* by applying a physical H-gate to each physical data qubit. However, this changes the positions of the *X*- and *Z*-stabilizers of the logical qubit, which can be considered as if the original logical qubit had been rotated by 90°. Consequently, performing joint measurement-based logical CNOT operations between adjacent logical qubits is challenging [27, 30–32]. Gate teleportation is a technique used to transmit the state of a logical qubit from one location to another within a lattice while simultaneously performing a Hadamard transformation. This method uses a specialized *ZX* joint measurement [33, 34] that projects the shared boundary stabilizers of two logical qubits onto the *ZX* plane of the surface code lattice. However, this method does not preserve the original shape of the logical qubit as in the transversal approach. Herein, shape refers to the geometry of the regular lattice used to visualize the stabilizers. Another approach involves the use of code deformation techniques to produce logical-*H*. This method expands and shrinks the code space of a logical qubit in the desired direction to generate logical-*H* while preserving the original shape of the logical qubit. However, this approach requires extensive coding.

This study guides the implementation of logical-*H*s while preserving the integrity of logical qubits. First, the transversal logical-*H* is applied to a logical qubit. However, this changes the original shape of the logical qubit as determined by the configuration of the boundary stabilizer. To address this problem, we propose a new method called *boundary deformation*, which relies only on a few boundary stabilizers and physical qubits to restore the shape of a logical qubit. This approach is more space-efficient than conventional methods and provides a logical-*H* that preserves the desired shape.

The remainder of this paper is organized as follows: Section 2 presents a literature review introducing the concept of logical-*H*s in rotated surface codes. The rationale, behavior, and advantages of these techniques are also discussed. Section 3 elaborates on the proposed method for generating logical-*H*s via boundary deformation and provides a detailed cost analysis demonstrating how much less code space is required. Finally, we conclude the paper in Section 4.

2 | RELATED WORKS

The rotated surface code is a variant of the surface code, an extensively employed QEC code in quantum computing [15, 23, 26–31, 35, 36]. This code builds *X*- and *Z*-stabilizers on a two-dimensional lattice of qubits to detect and correct quantum errors. In the rotated surface code, the *X*- and *Z*-stabilizers are rotated by 45° along the lattice, resulting in their alignment along the edges of the diagonals rather than the lattice edges. This rotation facilitates a more compact arrangement of qubits, thereby reducing the overall code overhead. The level of fault tolerance that can be achieved using the rotated surface code depends on the code distance. The code distance refers to the minimum number of Pauli operations that can be executed consecutively between facing boundaries without errors. For example, a rotated logical qubit with code distance d is a $d \times d$ lattice consisting of d^2 data qubits and d^2-1 ancilla qubits. The total number of physical qubits required was $\approx 2d^2$.

In the rotated surface code, the *X*- and *Z*-stabilizers encode the quantum state of a logical qubit in a unique bit string format called a codeword. The logical quantum state $|\Psi_L\rangle$ is obtained by projecting it onto codewords of the $(+)$ eigenspace of all stabilizers, as shown in (1),

$$|\psi_L\rangle = \frac{1}{N}\prod_{S_i \in \langle S\rangle, 0\langle i\langle n}\left(I^{\bigotimes n} + S_i\right)|\psi^{\bigotimes n}\rangle, \qquad (1)$$

where S is the stabilizer group comprising S_i (*X*- or *Z*-stabilizer), n is the number of data qubits, i is the index of the ancilla qubit, and N is the normalization factor [37]. The elements in Group S included weight-2 or weight-4 stabilizers. The former, called a boundary stabilizer, detects errors between two neighboring data qubits on a lattice boundary, whereas the latter detects errors in four adjacent data qubits within the lattice.

In the study of Fu [30], a rotated surface code was used to identify four logical qubit flavors based on boundary stabilizers placed at the bottom of the lattice: *X*_Right, *X*_Left, *Z*_Right, and *Z*_Left, as illustrated in Figure 1A. Lattice surgery [15, 23, 27, 30–32], which entails joint measurements between adjacent logical qubits, relies on flavor. Figure 1B,C illustrates the lattice operations of the two logical qubits using the boundary stabilizers. In Figure 1B, two logical qubits with adjacent *X*-stabilizers are merged into a logical qubit flavor, as shown on the right side. The logical *X* operator on the merged logical qubit extends along the neighboring *X* stabilizers, indicating that lattice surgery can be performed by modifying the boundary stabilizers connecting the two logical qubits. In contrast, Figure 1C,D illustrates a case in which it is impossible to perform lattice surgery for a logical CNOT operation between neighboring logical qubits. The $Z+X$ boundary stabilizers link two logical qubits. Therefore, to facilitate lattice surgery

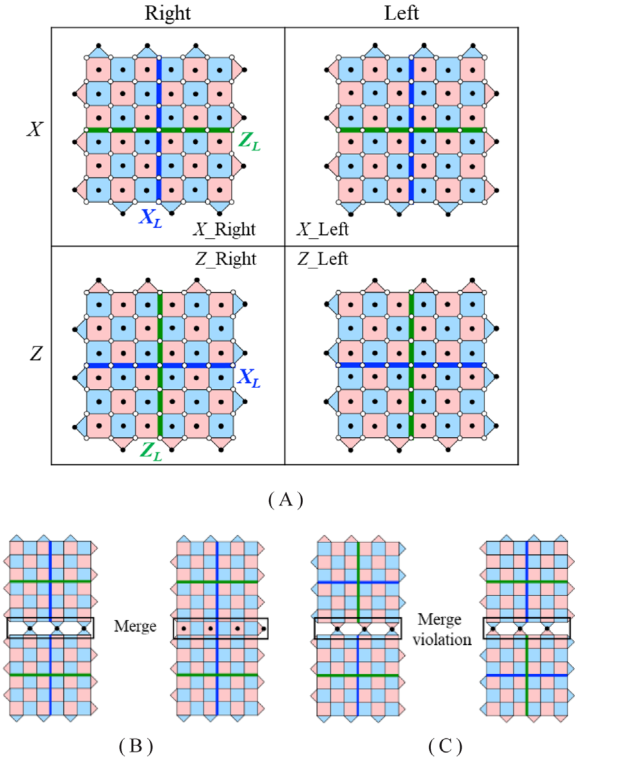

FIGURE 1 Four flavors for rotated surface code by a distance d-7. The figure shows data qubits as open circles and ancilla qubits as filled circles, with the blue and pink faces representing the X- and Z-stabilizers, respectively. (A) The flavor of a logical qubit is defined by a specific combination of the position and type of the boundary stabilizers. The boundary stabilizer at the bottom of the lattice identifies the flavor type. It is labeled X_* if the X stabilizer is on the bottom and Z_* otherwise. It is also labeled *_Left if the stabilizer starts as the leftmost corner of the lattice and *_Right otherwise. The bold blue and green lines represent the logical X and Z operators (X_L, Z_L), respectively, which are flavor-dependent. (B) Example of merging two logical qubits using facing boundary stabilizers. For an optimal merge operation, boundary stabilizers of the same type (X or Z) must face each other. (C) Example showing the inability to merge two logical qubits due to a boundary stabilizer type mismatch.

between these qubits, the qubits should be modified to guarantee an $X+X$ or $Z+Z$ boundary stabilizer configuration.

The following sections discuss the methodologies for implementing logical-Hs in rotated surface code.

2.1 | Transversal logical-H

Transversal logical-H in the surface code involves the application of a sequence of physical H-gates to each data qubit within a two-dimensional lattice. This transformation modifies the encoded state while preserving the surface code error correction capabilities.

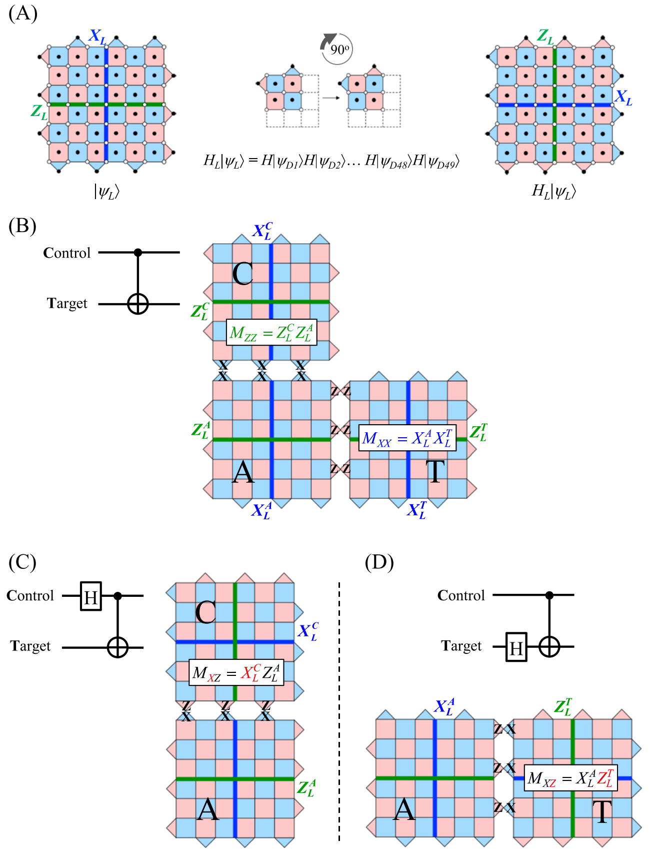

FIGURE 2 (A) Demonstration of logical-H applied using the transversal method. (B) Layout of logical qubits for a logical Controlled-NOT (CNOT) based on the lattice surgery. "C" stands for control qubit, "A" denotes the ancilla qubit, and "T" is the target qubit. M_{ZZ} (M_{XX}) denotes the joint measurement of the two logical Z(X) operators. (C)–(D) Examples in which lattice surgery operations are impossible due to the type mismatch of boundary stabilizers after logical-H. M_{XZ} denotes the joint measurement of the logical X and Z operators, transferring the current state to the Hadamard-transformed state.

Figure 2A illustrates that the quantum state of the logical qubit is transformed from $|\Psi_L\rangle$ to the Hadamard-transformed $H_L|\Psi_L\rangle$, giving the impression that the flavor of the logical qubit is rotated to the right by 90°. However, this method has a major disadvantage in that it exchanges stabilizer generators for the logical X and Z operators. Thus, when the transversal logical-H is complete, the positions of the logical X(Z) operators (X_L, Z_L) and X(Z) stabilizers are interchanged. From the perspective of logical qubit flavor, this transition corresponds to a transformation from X_Left to Z_Left.

Figure 2B illustrates the lattice surgery-based logical CNOT circuit. In this circuit, three logical qubits are considered: C (control), T (target), and A (ancilla). The objective was to implement a logical CNOT operation by merging the boundary stabilizers. To achieve this, C can transfer logical quantum bits to T via lattice surgery using an intermediate logical qubit A, and T can transfer logical quantum phases to C. The order in which the logical qubits undergo lattice surgery is irrelevant; both C-A-T and T-A-C are valid.

It is crucial to connect adjacent boundary stabilizers of the same type. As illustrated in Figure 2B, C and A should be connected to the X-stabilizers to enable the M_{ZZ} operation ($Z_L^C Z_L^A$), which merges two logical qubits, that is, the logical X operators of C and A are extended to the merged logical qubit. Only the M_{XX} operation ($X_L^A X_L^T$) is required for A and T connected to the Z-stabilizers; however, the underlying principle remains the same. Thus, it is evident that logical qubits coupled with the same boundary stabilizers can perform logical CNOT via joint measurement-based lattice surgery.

Applying transverse logical-H to the lattice illustrated in Figure 2B violates the joint measurement rule described earlier. This is due to the characteristics of the logical qubit change. Figure 2C,D illustrates the application of transversal logical-H to C or T. As illustrated in Figure 2C, the type C boundary stabilizer transforms from X to Z during the merging process. This results in a mismatch of the stabilizer type, which makes the M_{ZZ} operation difficult. Similarly, in Figure 2D, the boundary stabilizer type T is modified from Z to X, thus preventing the M_{XX} operation between A and T.

In summary, although transversal logical-Hs can transform logical quantum states, they can cause a delay in logical CNOT operations through lattice surgery by altering the geometry of the logical qubit. Consequently, additional steps are required to restore the original flavor of the logical qubit after applying transversal logical-Hs.

2.2 | Gate teleportation

Logical gates can be created using the principle of quantum teleportation between two logical qubits [22, 23]. This technique allows the transmission of a logical quantum state without explicitly transmitting the physical state; that is, the state of one logical qubit is teleported to another using a pair of Bell states. This process requires the execution of a joint measurement and compensation for the state of the receiving qubit, based on the measurement outcomes of the sending qubit. Figure 3A illustrates

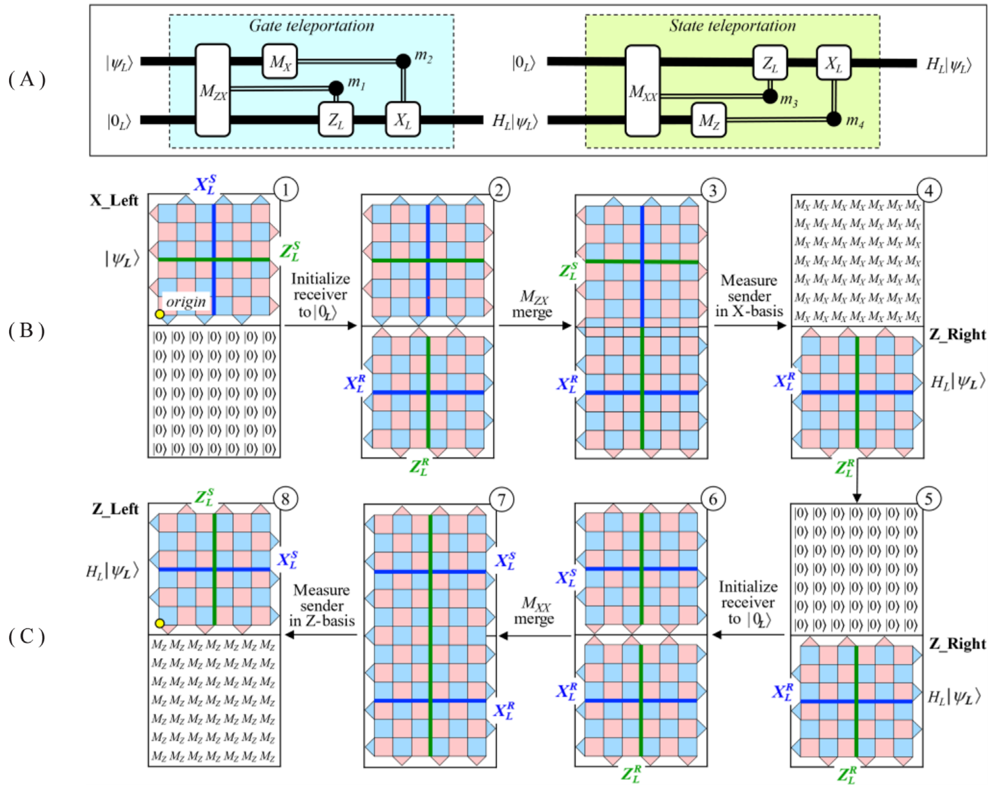

FIGURE 3 Logical-H by the teleportation protocols between two logical qubits, a sender and a receiver. (A) Lattice surgery implementation. This approach consists of two steps: gate teleportation and state teleportation. Thick lines represent logical qubits. Double lines are classical bits ($m_i = \{0, 1\}$) indicating measurement outcomes. (B) Surface code implementation. The sender's logical state, $|\Psi_L\rangle$, is transmitted to the receiver as $H_L|\Psi_L\rangle$ by the M_{ZX} operation between heterogeneous boundary stabilizers. The flavor of the logical qubit is transformed from X_Left to Z_Right, which is the same result as that of the transversal method. (C) The receiver's logical state can be returned to the origin (yellow circle) by the M_{XX} operation; however, the original flavor of the logical qubit is not restored.

the lattice surgery implementation for logical-H. It consists of two steps: (i) teleportation of logical-H and (ii) retransmission of the Hadamard-transformed logical state back to its original position.

Figure 3B illustrates the gate teleportation method using surface codes. The ZX joint measurements are essential for gate teleportation and state transfer. This unique joint measurement method projects the shared boundary stabilizers of two logical qubits onto the ZX plane of the surface code [33, 34]. If the sender's flavor for the ZX joint measurement was X_Left, then the receiver's flavor was Z_Right. The joint measurement incorporated weight-2 X-stabilizers at the bottom of the sender and weight-2 Z-stabilizers at the top of the receiver into the weight-4 XZ-stabilizers. This process temporarily created weight-4 ZX stabilizers adjacent to the unified stabilizers. The product of these novel stabilizer measurements determines the value of m_1, and the product of the X-basis measurements (M_X) of all sender data qubits determines the value of m_2. The two values compensate for the isolated receiver: The logical Z operator (Z_L^R) is applied if $m_1 = 1$, and the logical X operator (X_L^R) is applied if $m_2 = 1$. Thus, the sender's logical state $|\Psi_L\rangle$ is transmitted to the receiver as $H_L|\Psi_L\rangle$. However, gate teleportation does not guarantee the preservation of the original flavor of the logical qubit. As illustrated in Figure 3, the flavor was transformed from X_Left to Z_Right, yielding the same result as the transversal method described in Section 2.1.

In the second step, state teleportation (ST) is used to return the Hadamard-transformed qubit to its original position. Depending on the flavor of the qubit, this can be accomplished through a series of lattice surgery operations using either M_{XX} or M_{ZZ} measurements. The process of merging, splitting, and compensating the quantum states using the measurement outcomes (m_3 and m_4) is illustrated in Figure 3C and is similar to the process illustrated in Figure 3B.

Following ST, the flavor changed from Z_Right to Z_Left, which differed from the initial flavor X_Left, as illustrated in Figure 3B. This method guarantees the evolution of logical quantum states by using logical-Hs. However, this does not preserve the flavor of the initial qubit. Therefore, gate teleportation requires a step to restore flavor, which increases space requirements.

2.3 | Code deformation

Code deformation is a powerful method used to control the geometry of a surface code while maintaining its error detection and correction capabilities [24–29]. The flexible operation of logical qubits in a two-dimensional lattice is made possible by expanding the lattice in one direction and then contracting it back. Two methods exist for implementing a logical-H; these preserve the intrinsic properties of a logical qubit. The first method implements a logical-H that fully preserves all properties; however, it necessitates six additional ancillary patches [27]. The second method implements logical-H by adding three ancillary patches but requires additional steps, such as ST, to return to the initial flavor [28, 29]. Although the second method requires less additional space than the first, it increases the space cost because most operations are occupied and executed in all patches. Consequently, the difference in the cost of a single logical-H between the two methods was negligible. In addition, the cost of running multiple logical-Hs on a checkerboard, as discussed in Section 3.2.2, is higher for the second method because it requires more move operations than the first. Therefore, we focused on the first method, which was used in this study. Figure 4A illustrates the lattice surgery implementation of the code deformation for logical-H, which consists of three phases: (i) transversal logical-H, (ii) logical-operator rotation via code deformation, and (iii) the return of the logical qubit to its original position. Finally, by measuring the $2d^2 + d$ data qubits, the original size of the logical qubit with the initial flavor was restored.

Figure 4B illustrates the implementation of the surface code. This method requires six additional patches, each occupied by a single lattice. The primary goal is to implement a logical-H that preserves the flavor of a qubit at its exact location. The initial state of the logical qubit is presented in Step 1; in Steps 2 and 3, the transversal logical-H transforms the flavor from X_Left to Z_Left. Steps 4 and 5 expand the logical qubit in the desired direction, restoring it to the X_Left flavor while maintaining its state by measurement. Growing the logical qubit while measuring on the Z- and X-bases returns it to its original position in Steps 6–9. Most importantly, the flavors in Steps 1 and 9 remained the same.

This technique facilitates logical-H while preserving the unique properties of the qubit. However, this requires considerable code space. As illustrated in Figure 4, the logical-H technique requires six additional patches.

3 | METHODS

This study aims to reduce the storage space required for logical-Hs in rotated surface codes. As discussed in Section 2.1, applying transversal H-gates to data qubits transforms the flavor of the logical qubit, thereby changing the definition of the logical operators. In general, lattice surgery allows logical operations between qubits

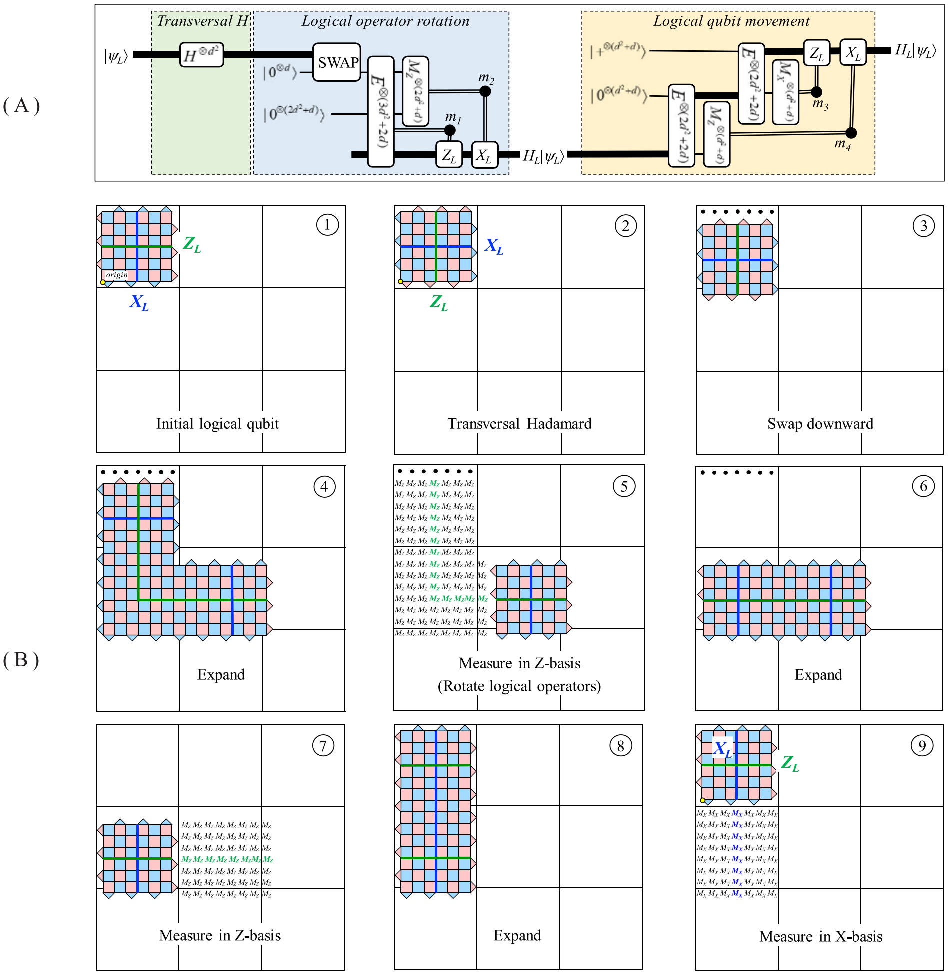

FIGURE 4 Logical-*H* by code deformation method. (A) Lattice surgery implementation. Thick lines indicate logical qubits. Thin lines indicate individual physical qubits. $A^{\otimes n}$ represents the tensor product of the A operator applied to each of the n physical qubits, where the operators include initialization ($|0\rangle$, $|+\rangle$), Hadamard (H), and Z-basis and X-basis measurements (M_Z, M_X). $E^{\otimes n}$ denotes the expansion of a logical qubit up to n qubits by adding physical qubits. (B) Surface code implementation. This method requires seven patches for the lattice.

aligned with the same type of boundary stabilizer. However, as mentioned in Section 2.3, restoring the original logical operator definitions and qubit flavors requires considerable code space. We propose herein a novel approach referred to as the "boundary deformation-based logical-*H* operation." This method does not require additional space and supports logical operations with fewer than two patches.

3.1 | Boundary deformation-based logical-*H* operation

The boundary deformation technique restores the logical qubit flavor and operator definitions. Figure 5 illustrates this method using the rotated surface code by the distance *d*-7.

Figure 5A depicts the initial logical qubit prepared for the boundary deformation process, assuming that the transversal logical-*H* was applied in advance. This qubit represents the *Z*_Left flavor transformed by transversal logical-*H* from the *X*_Left flavor, which is the state $|\Psi_L\rangle$, as illustrated in Figure 2A. The Hadamard-transformed logical qubit was assumed to be in the simultaneous +1-eigenstate of all the stabilizers and was denoted as $|\psi_L^H\rangle$. Let us define the stabilizer group that determines the logical state as $S = \langle S_{\text{internal}}, S_{\text{old-boundary}}\rangle$, where S_{internal} consists of weight-4 stabilizers inside the qubit, and the $S_{\text{old-boundary}}$ comprises weight-2 stabilizers on the boundary. These predeformation boundary stabilizers are denoted by $S_{\text{old-boundary}} = \langle Z_{Si}^H, X_{Sj}^H\rangle$, where $i, j \in \{1, 2, ..., d-1\}$. The logical Z and X operators that perform nontrivial operations are defined as Z_L^H and X_L^H, respectively.

Figure 5B illustrates *Z*_Left (from Figure 5A) being deformed into *X*_Right. The first step in the boundary deformation is to replace the $S_{\text{old-boundary}}$ of the stabilizer group with a new boundary set, $S_{\text{new-boundary}} = \langle X_{Si}^D, Z_{Sj}^D\rangle$. Herein, we assume that there is no measurement error in the stabilizer qubits throughout the *d*-round measurement cycles, as discussed in [14, 35]. When a new stabilizer anticommutes with an element in the stabilizer group, it replaces the anticommuting element in two ways, depending on the stabilizer measurement result: If the measurement result is +1, it will replace the anticommuting element directly; however, if the measurement result is −1, the new stabilizer replaces the

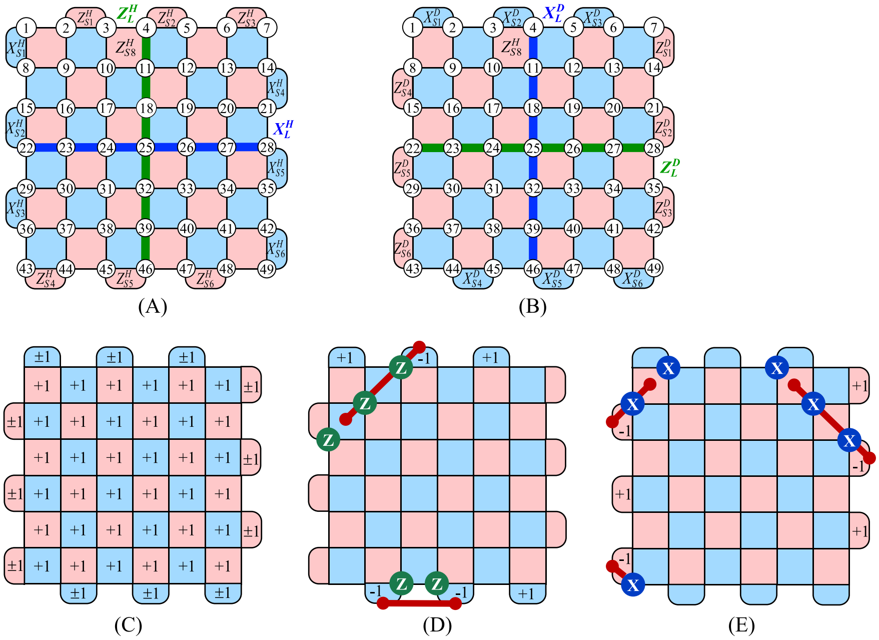

FIGURE 5 Boundary deformation in rotated surface code by a distance *d*-7. It comprises 49 data qubits (numbered circles) and 48 ancilla qubits (inside each face). The superscripts "*H*" and "*D*" stand for Hadamard and deformation, respectively. The subscript "*S**" represents the index of the stabilizer. (A) Initial logical qubit. This qubit represents the *Z*_Left flavor transformed by transversal logical-*H* from the *X*_Left flavor. (B) Boundary deformation to *X*_Right flavor. This process changes the flavor while preserving the quantum state of the logical qubit. (C) The outcomes of stabilizer measurements during the deformation process. The internal stabilizer consistently measures the same state, while the new boundary stabilizer measures a random ±1. (D) *Z*-syndrome stabilization detected during the deformation of the upper and lower boundaries. (E) *X*-syndrome stabilization detected during the deformation of the left and right boundaries.

anticommuting element only after it has changed to the $+1$ state by applying its conjugate partner operator [2, 26]. The commutativity of operators is based on the algebra of Pauli operators; *X* and *Z* operators always commute ($[\widehat{X}_a, \widehat{Z}_b] = 0$) on different qubits but always anticommute ($[\widehat{X}_a, \widehat{Z}_a] \neq 0$) on the same qubit [14, 38]. In the context of operator measurements and commutativity, it was observed that each element of the $S_{\text{new-boundary}}$ commutes with all those in the S_{internal} boundary but anticommutes with some of those in the $S_{\text{old-boundary}}$. For example, X_{S2}^D in the $S_{\text{new-boundary}}$ commutes with Z_{S8}^H in S_{internal} but anticommutes with Z_{S1}^H in the $S_{\text{old-boundary}}$ on data qubit 3, as shown by the following notation,

$$\begin{aligned}
\left[X_{S2}^D, Z_{S8}^H\right] &= \left[\widehat{X}_3\widehat{X}_4, \widehat{Z}_3\widehat{Z}_4\widehat{Z}_{10}\widehat{Z}_{11}\right] \\
&= \left[\widehat{Z}_3\widehat{Z}_4\widehat{Z}_{10}\widehat{Z}_{11}\widehat{X}_3\widehat{X}_4, \widehat{X}_3\widehat{X}_4\widehat{Z}_3\widehat{Z}_4\widehat{Z}_{10}\widehat{Z}_{11}\right] = 0, \\
\left[X_{S2}^D, Z_{S1}^H\right] &= \left[\widehat{X}_3\widehat{X}_4, \widehat{Z}_2\widehat{Z}_3\right] \\
&= \left[\widehat{Z}_2\widehat{Z}_3\widehat{X}_3\widehat{X}_4, \widehat{X}_3\widehat{X}_4\widehat{Z}_2\widehat{Z}_3\right] \neq 0.
\end{aligned}$$

If X_{S2}^D has a measurement outcome of $+1$, it replaces Z_{S1}^H in the old group. If the outcome is -1, it is corrected to $+1$ by applying the Pauli-*Z* operator followed by the replacement of Z_{S1}^H. In the same way, the remaining elements of the $S_{\text{new-boundary}}$ replace the anticommuting elements of the $S_{\text{old-boundary}}$, and the old group is reorganized as a new stabilizer group $S' = \langle S_{\text{internal}}, S_{\text{new-boundary}} \rangle$. In addition, it was observed that in data qubit 4, a new *X*-stabilizer X_{S2}^D and an old logical *Z* operator (Z_L^H) anticommute. Similarly, in data qubit 28, a new *Z*-stabilizer Z_{S5}^D and an old logical *X*-operator (X_L^H) anticommute. Consequently, the operators can no longer act on a deformed logical qubit. The postmeasurement state ($|\psi_L^M\rangle$) was projected (ignoring normalization) to

$$|\psi_{\pm L}^M\rangle = \prod\nolimits_i (I \pm M_i)\,|\psi_L^H\rangle, \tag{2}$$

where $M_i \in S_{\text{new-boundary}}$ and the sign ± denote the measurement outcomes of the $S_{\text{new-boundary}}$ elements anticommuting with the $S_{\text{old-boundary}}$. These stabilizers were fixed to $+1$-eigenstates in the subsequent step.

The second step in boundary deformation is identifying and stabilizing new syndromes depending on the measurement outcomes. If all the outcomes are $+1$, the logical qubit is in the state $|\psi_{+L}^M\rangle$, which is the same logical quantum state as in Figure 5A. However, if any outcomes are -1, the logical qubit is in the state $|\psi_{-L}^M\rangle$, and a decoding algorithm, such as minimum weight perfect matching (MWPM) [39–41], is used to deduce the shortest correction path. As $X^2 = Z^2 = I$, all syndromes are corrected by applying a Pauli-X or -Z operator to each physical data qubit along the correction path. This correction process transitions the states of several physical qubits to allow all stabilizers to return to their $+1$-eigenstates. Odd physical state transitions across the logical qubit induce a logical bit or phase flip. This state is compensated by applying the newly defined logical X or Z operators as postprocessing operators. Consequently, the postdeformation state can preserve the initial encoding state in the form of (2) based on the following steps: (i) measurement into the deformed group (S'), (ii) stabilization with Pauli correction, and (iii) postprocessing with logical operators.

$$|\psi_L^D\rangle = \left(Z_L^D\right)^{(1-m_z)/2}\left(X_L^D\right)^{(1-m_x)/2}|\psi_L^M\rangle. \tag{3}$$

In (3), X_L^D and Z_L^D are the logical X and Z operators newly defined in the deformed logical qubit. Two key factors determine the postprocessing correction: m_z and m_x, where m_z is the parity of the phase-flipped qubits by the Pauli-Z operator, and m_x is the parity of the bit-flipped qubits by the Pauli-X operator. Each had a $+1$ (even) or -1 (odd) value. Refer to Figure 5C–E for examples to understand this concept.

Figure 5C illustrates the results of the stabilizer measurements during the deformation process. In this example, the internal stabilizer maintains a consistent state, whereas the new boundary stabilizer randomly performs $+1$ or -1 measurements. Figure 5D illustrates the Z-syndrome stabilization process detected during the deformation of the upper and lower boundaries. This demonstrates the shortest correction path inferred for the Z-syndromes detected by X-stabilizers X_{S2}^D, X_{S4}^D, and X_{S5}^D. The syndrome in X_{S2}^D was stabilized by applying the Pauli-Z operator to data qubits 3, 9, and 15, whereas the syndromes in X_{S4}^D and X_{S5}^D were stabilized by applying the Pauli-Z operator to data qubits 45 and 46. During the correction, an odd number of phase flips in the data qubits affected the logical phase ($m_z = -1$). This was corrected by applying a newly defined logical Z-operator. Figure 5E illustrates how the X syndrome is stabilized when the left and right boundaries are deformed. In this example, the Z-stabilizers Z_{S1}^D, Z_{S3}^D, and Z_{S5}^D detect the X syndrome, and the shortest correction path involves the application of the Pauli-X operator to data qubits 2, 5, 8, 13, 21, and 43 to stabilize them. These corrections cause bit flips in the data qubits that construct the logical qubit but do not affect the state of the logical qubit ($m_x = +1$). By substituting the m_z and m_x values generated in Figure 5D,E into (3), the logical quantum state can be summarized as $|\psi_L^D\rangle = \left(Z_L^D\right)^1\left(X_L^D\right)^0|\psi_L^M\rangle$. The initial logical state is preserved when a new logical Z-operator is applied to the deformed logical qubit. The classical *Pauli frame* method [42, 43] can handle all Pauli operations involved in the proposed approach, meaning that the execution overhead is negligible.

Figure 6 illustrates all the steps of logical-H that preserve the logical information and flavor of the logical qubit. It involves three main steps: the transversal of logical-H, rotation of logical operators via boundary deformation, and logical flip-and-shift. Cost estimation was performed using only two adjacent patches to enhance the observations. In both lattice spaces, the orientation of the ancillary region relative to the logical qubit can be up-, down-, left-, or right-handed. This example assumes the use of ancillary space to the right of a logical qubit. Figure 6A illustrates the implementation of the three aforementioned steps above using lattice surgery. Figure 6B illustrates the first two steps with the surface code implementation, and Figure 6C illustrates the last step.

Figure 6B presents the results after the application of logical-H to an initial logical qubit of the X_Left flavor containing the right ancillary region. The logical operators on the qubit are defined as X_L and Z_L. We applied the transversal logical-H described in Section 2.1 for the initial qubit. However, this led to changes in the stabilizer positions that resulted in changes to the definitions of logical operators ($X_L Z_L \rightarrow X_L^H Z_L^H$) and flavor ($X$_Left → Z_Left). The traditional code deformation technique described in Section 2.3 requires considerable space to restore the original flavor. By contrast, the proposed method can reduce the number of managed lattices to less than two to modify the flavor with boundary deformation. The logical phase or bit transition is identified and compensated in the correction operations for syndrome stabilization to ensure the integrity of the logical qubit state.

The deformed logical qubit restores the original logical operators but does not return to the original flavor. Therefore, we restored the original flavor of the logical qubit using a logical flip-and-shift process, as illustrated in Figure 6C. This operation flips the qubit horizontally, restores its original flavor, and shifts it back to its original position [26]. Two types of flip operations exist, namely, X and Z. The boundary of the logical qubit determines the type of flip operation used. The X-flip operation flips the flavor along the X-boundary using the physical qubits of the initial state $|+\rangle$. A Z-flip operation was used to flip the flavor along the Z-boundary. The process is illustrated in Figure 6C and involves the following steps: (i) preparation of the physical data qubits in the first

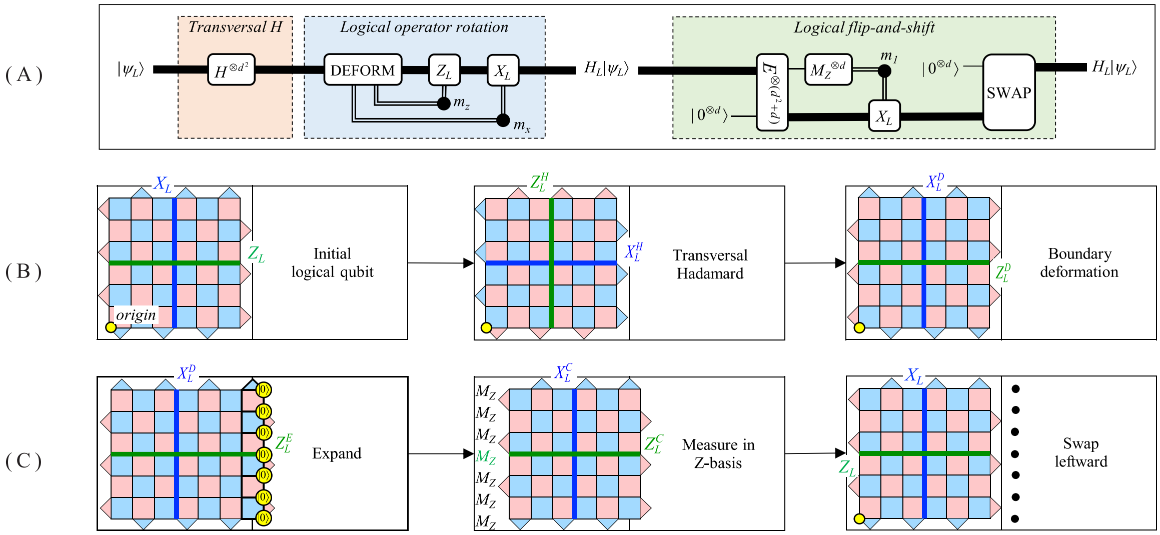

FIGURE 6 Logical-H by the boundary deformation on a rotated code by distance d-7. (A) Lattice surgery implementation. Thick lines indicate logical qubits. Thin lines indicate individual physical qubits. Double lines ending in black dots represent classical bits indicating measurement outcomes $m_i = \{0, 1\}$. Logical Pauli corrections need to be applied; these are conditioned on the measurement outcomes as $(P_L)^{mi}$. *DEFORM* involves the measurement of the deformed boundary stabilizers and stabilization of the resulting syndromes. $A^{\bigotimes n}A$ represents the tensor product of the A operator applied to each of the n physical qubits, where the operators include Hadamard (H) and Z-basis measurements (M_Z). $E^{\bigotimes n}$ represents the expansion of a logical qubit to n qubits by adding physical qubits. (B) Surface code implementation of boundary deformation. The initial state of the logical qubit with X_Left flavor. After transversal H, the flavor changes to Z_Left. By deforming the boundary stabilizers, the flavor turns up X_Right. (C) Surface code implementation of logical flip-and-shift. A $d \times d$ lattice transformed to an X_Left flavor by a logical Z-flip back to the origin (yellow circle) using swap operations.

column of the ancillary region adjacent to the right Z-boundary to state $|0\rangle$. Subsequently, the stabilizers were measured to expand it to the $d \times (d+1)$ code. (ii) Measurements of the physical data qubits in the first column of the expanded logical qubit into a Z-basis and contracts it to $d \times d$ code. The product of the eigenvalues (green, M_Z) measured using the logical Z operator (Z_L^E) determines the value of m_1 in Figure 6A. If $m_1 = -1$, the logical X operator (X_L^C) was applied to the contracted logical qubit to correct the logical bit. (iii) Shifting of the logical qubit to its original position via swap operations. The X-flip operation is handled similarly to the Z-flip operation, except that the bases of the added physical qubits were different.

3.2 | Comparison of space overhead

When performing a logical operation with a regular surface code for a distance d, the overhead is estimated as O(d^2) (space cost) and $O(d)$ (time cost) [44]. The space cost is determined by the number of physical qubits, where a single qubit includes both the data and ancilla qubits. The number of rounds of stabilizer measurements determines the time cost. The overhead of any logical operation is expressed as the space–time cost in $O(d^3)$ units. If a logical operation uses p patches for t time steps, the space–time cost is $pt{\cdot}d^3$ and the space cost is $p{\cdot}d^2$, which is the number of physical qubits [36].

Several assumptions are postulated to estimate the space–time cost of logical-H. First, all the Pauli operations were excluded from the cost analysis because they can be efficiently handled using classical computing methods. Second, the time cost did not account for processes that did not require stabilizer measurements. Third, the time costs of fault-tolerant transversal operations, such as physical qubit initialization, measurement, transversal H, and swapping, were not considered. Finally, the space cost was estimated by converting the number of patches to represent a logical qubit arrangement. A patch represents a regular grid space in which logical qubits can be placed on a two-dimensional physical qubit plane. It occupies only one logical qubit at a time, even if it is only partially used. The estimation of the space cost of a logical operation based solely on the size of a logical qubit can differ from the actual amount of physical space consumed. Therefore, we estimated the space cost of a logical operation as the number of patches used by a logical qubit. For example, if a logical qubit occupies part of another patch or is contained in two patches, the estimated space requirement is calculated as $2d^2$.

In addition, logical operations distort the width of a logical qubit in a regular lattice, as observed in the code deformation technique, which modifies the physical scale of a logical qubit by adding qubits to it. The space cost was determined by the number of patches required to perform operations using these unstructured codes. As shown in Figure 4B, when a logical qubit extends across seven patches, the space cost is approximately seven times that of conventional code.

In this study, we analyze the cost of single and multiple logical-H using the code deformation-based method in Section 2.3 and the boundary deformation-based method in Section 3. The gate teleportation method described in Section 2.2 was excluded from our analysis because it yielded the same results as those yielded by the transversal logical-H.

3.2.1 | Single logical-H operation

The cost of a single logical-H was evaluated as the sum of all the operations performed by the circuits shown in Figures 4 and 6. The implementation costs of both techniques are shown in Figure 7 and summarized in Table 1.

Figure 7A presents a comprehensive cost estimate for the code formation-based technique shown in

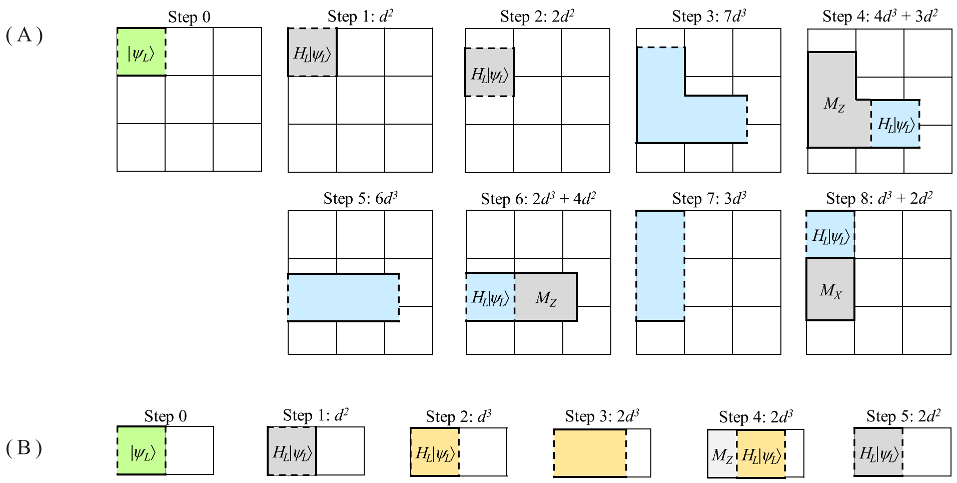

FIGURE 7 The space–time cost associated with the execution of a single logical-H. The green lattice represents operations that prepare a logical qubit in an arbitrary state. The blue and yellow lattices represent operations with an estimated space–time cost. The gray lattice represents operations with only a space cost, including initialization, measurement, transversal H, and swap operations. The space cost is estimated as the number of patches a logical qubit occupies. The dashed lines in a lattice represent the Z-boundary, and the straight lines represent the X-boundary. M_P denotes the P-basis measurement of data qubits. (A) Code deformation-based logical-H. The total space–time cost is $23d^3 + 12d^2$. (B) Boundary deformation-based logical-H. The total space–time cost is $5d^3 + 3d^2$.

TABLE 1 Comparison of logical operation costs in two methods. CD denotes *code deformation* (Figures 4 and 7A) and BD denotes *boundary deformation* (Figures 6 and 7B).

	Operation (maximum patches)		**Space–time cost**		**Space cost**	
Step	**CD**	**BD**	**CD**	**BD**	**CD**	**BD**
1	Transversal H	Transversal H	-	-	d^2	d^2
2	Swap	BD	-	d^3	$2d^2$	-
3	Expand (7)	Expand (2)	$7d^3$	$2d^3$	-	-
4	M_Z and shrink	M_Z and shrink	$4d^3$	$2d^3$	$3d^2$	-
5	Expand (6)	Swap	$6d^3$	-	-	$2d^2$
6	M_Z and shrink	-	$2d^3$	-	$4d^2$	-
7	Expand (3)	-	$3d^3$	-	-	-
8	M_X and shrink	-	d^3	-	$2d^2$	-
Total			$23d^3$	$5d^3$	$12d^2$	$3d^2$

Figure 4. The space cost (excluding the time cost) for Steps 1 and 2 is estimated by considering the sequential execution of transversal H and swap operations. In Step 2, one lattice is present in both patches; hence, the total space required is $2d^2$. The expansion cost estimates for Steps 3, 5, and 7 consider the initialization of additional physical qubits. The predicted space–time costs were $7d^3$, $6d^3$, and $3d^3$, respectively, considering the number of patches occupied by a single lattice and stabilizer measurements. The measurement operations (M_X, M_Z) contributed to the space costs of Steps 4, 6, and 8 with estimated space requirements of $3d^2$, $4d^2$, and $2d^2$, respectively. The space–time costs of the shrink operations, including four, two, and one patches, and stabilizer measurements in each step, were estimated to be $4d^3 + 3d^2$, $2d^3 + 4d^2$, and $d^3 + 2d^2$, respectively. The total estimated cost was $23d^3 + 12d^2$. Figure 7B provides a comprehensive estimate of the costs associated with the boundary deformation technique, as shown in Figure 6. In this process, Steps 1 and 5 involve the transversal H and swap operations that determine the space cost based on the number of patches, excluding the time cost. Because all the operations occur in one patch, d^2 defines the space requirements. In Step 2, the cost of the boundary deformation is determined by the cost of the stabilizer measurements in the same patch. A single patch required d^3 to measure the lattice. In Step 3, the expansion operation required the initialization of additional physical qubits at a cost of $2d^3$ because the expanded lattice comprised two patches and required stabilizer measurements. In the shrink operation in Step 4, the measurement operation (M_Z) and associated space cost are included in the ultimate cost, which is computed as $2d^3$. This technique approximates the total space–time cost to $5d^3 + 3d^2$. The two methods resulted in cost savings of approximately four times, regardless of the distance.

3.2.2 | Multiple logical-*H* operations

The cost of multiple logical-*H*s was determined by executing the circuit shown in Figures 4 and 6 parallel to several logical qubits.

The logical qubit layout assumed a checkerboard structure [31]. This structure is advantageous because it provides an optimal space for implementing lattice surgery-based logical operations by creating an ancillary space between two data qubits and placing them adjacent to each other at the same *X*- or *Z*-boundary.

The experimental quantum circuit and logical qubit layout for the cost analyses are shown in Figure 8A,B. Eight logical data qubits and the same number of ancillary spaces are arranged in a 4 × 4 grid. First, we examined the execution of the circuit shown in Figure 8A using a code formation technique. Logical-*H* requires six additional contiguous ancillary spaces in the circuit, as shown in Figure 7A. However, only one adjacent continuous ancillary space was allowed. To create sufficient space, it was necessary to relocate the colliding logical qubits temporarily. This was achieved using the ST protocol, as shown in Figure 3C. Figure 8C shows the entire circuit implementation in Figure 8A, including the move operations based on STs. This is an example of a configurable circuit in which LQ1 is the starting point of execution. *ST* represents two moves: evacuation of a logical qubit to an ancillary space (dashed box) and transformation to its original location (solid box). A layer represents a set of logical-*H*s that operate simultaneously. If a certain logical-*H* required move operations, it was grouped into the same layer. The first layer in Figure 8C consists of a logical-*H* after moving LQ3 through LQ6. Figure 8E shows an extended LQ1 occupying the space vacated by the movement of the three logical qubits in the first layer during the execution of logical-*H*. The seven lattice cells were extended by the data spaces LQ1, LQ3, LQ5, and LQ6, and the ancillary spaces connecting them. LQ3 and LQ4 were relocated to the upper ancillary region, whereas LQ5 and LQ6 were relocated to the lower ancillary region. After relocation, logical-*H* was executed. Similarly, in Figure 8C, the logical-*H*s from LQ2 to LQ8 are executed sequentially in Layers 2–8. Finally, the four logical qubits that move to the ancillary regions are returned to their original locations.

Based on the execution steps shown in Figure 3C, the space–time cost of the state teleportation protocol was estimated to be $4d^3 + d^2$. This includes the costs of receiver initialization (d^3), merging ($2d^3$), and split-and-sender measurements ($d^3 + d^2$). To estimate the cost of a single logical-*H*, refer to the estimates in Section 3.2.1. There were eight moves and eight logical-*H*s in the circuit. Thus, the total space–time cost was calculated as $(23d^3 + 12d^2) \cdot 8 + (4d^3 + d^2) \cdot 8 = 216d^3 + 104d^2$.

The circuit shown in Figure 8A was executed using the proposed technique. This method allows logical operations to be executed simultaneously on different logical qubits, as long as they do not share physical data qubits that store computational information. Figure 8F shows that logical-*H*s are simultaneously conducted on eight logical qubits (LQ1–LQ8) without any shared physical data qubits. Consequently, each process was completed simultaneously, as shown in Figure 8D. The estimated total space–time cost was $(5d^3 + 3d^2) \cdot 8 = 40d^3 + 24d^2$ based on Section 3.2.1.

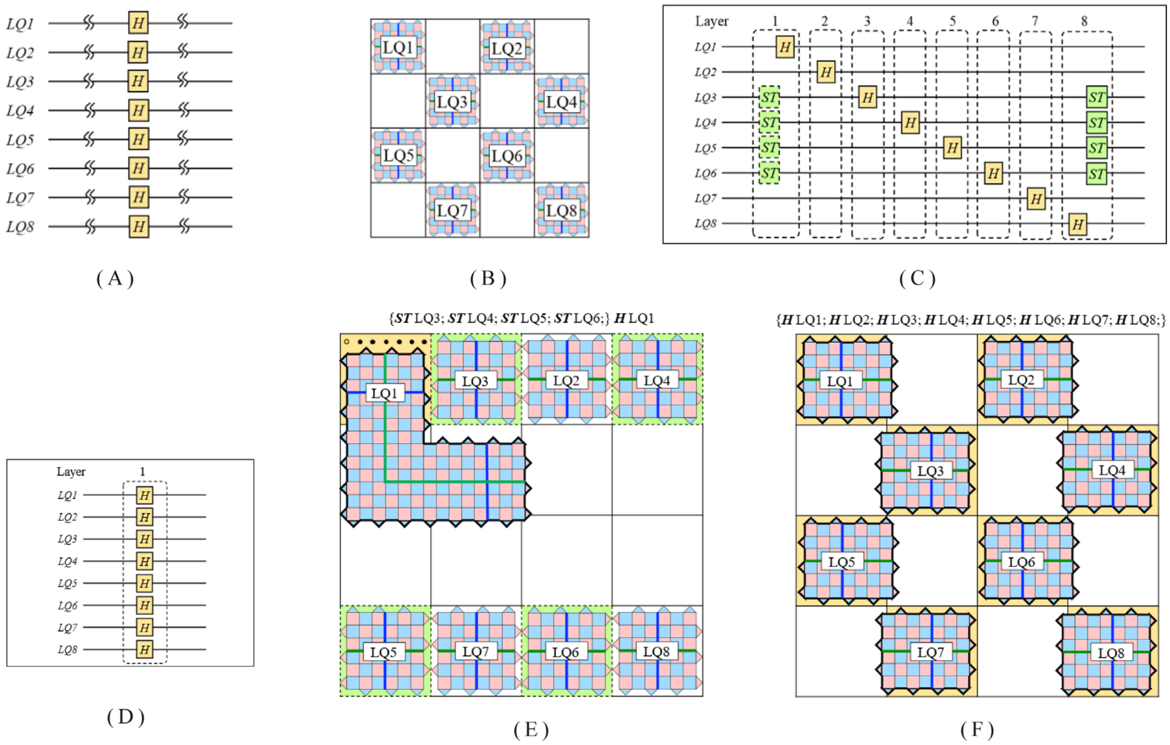
LQ1
LQ2
LQ3
LQ4
LQ5
LQ6
LQ7
LQ8
H
(A)
LQ1
LQ2
LQ3
LQ4
LQ5
LQ6
LQ7
LQ8
(B)
Layer
1
2
3
4
5
6
7
8
ST
H
(C)
Layer
1
(D)
{ST LQ3; ST LQ4; ST LQ5; ST LQ6;} H LQ1
(E)
{H LQ1; H LQ2; H LQ3; H LQ4; H LQ5; H LQ6; H LQ7; H LQ8;}
(F)

FIGURE 8 (A) Example of a quantum circuit with multi-parallel logical-*H*s. (B) Checkerboard-style logical qubit layout. (C) The procedure of the code deformation method. *ST* represents two moves: evacuation of a logical qubit to an ancillary space (dashed box), and transformation to its original location (solid box). A layer represents a set of logical-*H*s that operate simultaneously. If any certain logical-*H* requires move operations, it is grouped into the same layer. (D) The procedure of the boundary deformation method. The lattice with the bold lines in the yellow patch is in process. (E) Extended LQ1 to perform logical-*H* in code deformation method. The {} indicates parallelism. (F) Parallel logical-*H*s performed on eight logical qubits (LQ1–LQ8) without shared physical data qubits.

TABLE 2 Summary of simulation outcomes.

		Operation time	
Initial state	**Final state**	**Code deformation**	**Boundary deformation**
$\vert 0_L\rangle$	$\vert +_L\rangle$	141 ms	31 912 ms
$\vert 1_L\rangle$	$\vert -_L\rangle$	152 ms	32 017 ms
$\vert +_L\rangle$	$\vert 0_L\rangle$	159 ms	15 622 ms
$\vert -_L\rangle$	$\vert 1_L\rangle$	177 ms	15 493 ms

3.2.3 | Functional validation

These two methods were validated using QPlayer [45] and PyMatching [41]. QPlayer is a classical quantum simulator that supports a reduced Hilbert space, whereas PyMatching decodes QEC codes using the MWPM algorithm. The experiments were conducted on a Dell PowerEdge R740 server equipped with two Intel Xeon Gold 6132 central processing units (56 cores in total) and memory equal to 512 GB.

The validation process consisted of (i) initializing to a logical state, (ii) running logical-*H*, and (iii) measuring the final state on the distance-3 surface code. The syndrome measurement outcomes were decoded using PyMatching. The measurement results and operation times for the four initial logical states are summarized in Table 2. The final state validated the functionality of the proposed method, and the operation time confirmed the efficiency estimated from the cost analysis.

4 | CONCLUSIONS

This study presented a novel technique for implementing logical-*H* on rotated surface codes while maintaining the original logical operators. The flavor rotation of logical qubits in the logical-*H* process is a challenging problem that requires excessive overhead. Although transversal and gate teleportation methods minimize space usage, *X*- and *Z*-stabilizer exchanges do not retain the original characteristics of the logical qubit. The code deformation method preserves the flavor of the logical qubit but requires six times the extra space compared with the transversal approach.

This study addressed two challenges. First, the technique of boundary deformation was proposed to rotate logical operators affected by the transversal Hadamard. This rotation was achieved without additional spaces. The boundary stabilizers surrounding the logical qubit were selectively modulated to simplify the implementation of logical-*H*s. Furthermore, the logical phases or bits that underwent syndrome stabilization during boundary deformation were preserved as postprocessing corrections. Correction operations performed to restore syndromes caused by boundary deformation and postprocessing operations to compensate for the logical quantum state were treated as classical processes with no additional overhead. The flavor of a logical qubit was restored using logical flip-and-shift operations that require only one adjacent patch.

As explained by the cost estimation, the proposed technique was superior to conventional techniques in terms of space usage. The code deformation method required substantial code space to restore the original flavor of the logical qubit. In addition to the target patch, six other patches were required to execute logical-*H*. The estimated space–time cost of the proposed technique for a single logical-*H* is $5d^3 + 3d^2$, whereas that of the code deformation method is $23d^3 + 12d^2$. Regardless of the distance, the cost difference between the two approaches was approximately fourfold. The proposed method is beneficial when each logical qubit has limited free space, such as in a checkerboard-style arrangement. This prevents space sharing between logical qubits during logical-*H*s and enables simultaneous operations on multiple logical qubits.

ACKNOWLEDGMENTS

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. 2020-0-00014, A Technology Development of Quantum OS for Fault-tolerant Logical Qubit Computing Environment).

CONFLICT OF INTEREST STATEMENT

The authors declare that there are no conflicts of interest.

ORCID

Sang-Min Lee https://orcid.org/0009-0007-4851-1044
Ki-Sung Jin https://orcid.org/0000-0002-1997-3019
Jin-Ho On https://orcid.org/0000-0001-7610-2204

REFERENCES

1. P. W. Shor, *Schemes for reducing decoherence in quantum computer memory*, Phys. Rev. A **52** (1995), no. 4, R2493–R2496.
2. D. Gottesman, *A theory of fault-tolerant quantum computation*, Phys. Rev. A **57** (1998), 127–137.
3. D. Gottesman, *Stabilizer codes and quantum error correction*, Ph.D. Thesis, Caltech, 1997.
4. M. H. Freedman and D. A. Meyer, *Projective plane and planar quantum codes*, Found. Comp. Math. **1** (2001), 325–332.
5. D. Poulin, *Stabilizer formalism for operator quantum error correction*, Phys. Rev. Lett. **95** (2005), no. 23, 230504.
6. A. M. Steane, *A tutorial on quantum error correction*, In *Proc. International School of Physics "Enrico Fermi," Course CLXII, "Quantum Computers, Algorithms and Chaos"*, G. Casati, D. L. Shepelyansky, P. Zoller (eds.), IOS Press, Amsterdam, 2006, 1–32.
7. D. Gottesman, *An introduction to quantum error correction and fault-tolerant quantum computation*, Quant. Info. Sci. Contr. Math., Proc. Symp. App. Math. **68** (2010), 513–588.
8. Y. Wang, *Quantum computation and quantum information*, Stat. Sci. **27** (2012), no. 3, 373–394.
9. S. J. Devitt, W. J. Munro, and K. Nemoto, *Quantum error correction for beginners*, Rep. Prog. Phys. **76** (2013), no. 7, 076001.
10. E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, *Topological quantum memory*, J. Math. Phys. **43** (2002), 4452–4505.
11. S. B. Bravyi, and A. Y. Kitaev, *Quantum codes on a lattice with boundary*, arXiv preprint, 1998. DOI 10.48550/arXiv.quant-ph/9811052
12. D. S. Wang, A. G. Fowler, and L. C. L. Hollenberg, *Surface code quantum computing with error rates over 1%*, Phys. Rev. A **83** (2011), 020302.
13. A. G. Fowler, A. M. Stephens, and P. Groszkowski, *High-threshold universal quantum computation on the surface code*, Phys. Rev. A **80** (2009), 052312.
14. A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, *Surface codes: towards practical large-scale quantum computation*, Phys. Rev. A **86** (2012), 032324.
15. D. Horsman, A. G. Fowler, S. Devitt, and R. Van Meter, *Surface code quantum computing by lattice surgery*, New J. Phys. **14** (2012), 123011.
16. B. M. Terhal, *Quantum error correction for quantum memories*, Rev. Mod. Phys. **87** (2015), 307–346.
17. B. J. Brown, K. Laubscher, M. S. Kesselring, and J. R. Wootton, *Poking holes and cutting corners to achieve Clifford Gates with the surface code*, Phys. Rev. X. **7** (2017), 021029.
18. T. Tomaru, C. Yoshimura, and H. Mizuno, *Surface code for low-density qubit array*, Sci. Rep. **12** (2022), no. 1, 12946.
19. B. Zeng, A. Cross, and I. L. Chuang, *Transversality versus universality for additive quantum codes*, Theory IEEE Trans. Info. **57** (2011), 6272–6284.
20. D. Chandra, Z. Babar, H. V. Nguyen, D. Alanis, P. Botsinis, S. X. Ng, and L. Hanzo, *Quantum topological error correction codes are capable of improving the performance of Clifford gates*, IEEE Access. **7** (2019), 121501–121529.
21. A. G. Fowler, *Low overhead surface code logical Hadamard*, Quantum Inf. Comput. **12** (2012), 970–982.
22. D. Gottesman and I. L. Chuang, *Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations*, Nature **402** (1999), 390–393.
23. A. Erhard, H. Poulsen Nautrup, M. Meth, L. Postler, R. Stricker, M. Stadler, V. Negnevitsky, M. Ringbauer, P. Schindler, H. J. Briegel, R. Blatt, N. Friis, and T. Monz,

Entangling logical qubits with lattice surgery, Nature **589** (2021), no. 7841, 220–224.

24. H. Bombin and M. A. Martin-Delgado, *Quantum measurements, and gates by code deformation*, J. Phys. A. **42** (2009), 095302.
25. H. Bombin, *Clifford gates by code deformation*, New J. Phys. **13** (2011), 043005.
26. C. Vuillot, L. Lao, B. Criger, C. García Almudéver, K. Bertels, and B. M. Terhal, *Code deformation and lattice surgery are gauge fixing*, New J. Phys. **21** (2019), 033028.
27. A.G. Fowler, and C. Gidney, *Low overhead quantum computation using lattice surgery*, arxiv preprint, 2018. DOI 10.48550/arXiv.1808.06709
28. M. Beverland, V. Kliuchnikov, and E. Schoute, *Surface code compilation via edge-disjoint paths*, PRX Quantum. **3** (2022), no. 2, 020342.
29. M. E. Beverland, S. Huang, and V. Kliuchnikov, *Fault tolerance of stabilizer channels*, arxiv preprint, 2024. DOI 10.48550/arXiv.2401.12017
30. X. Fu, L. Lao, K. Bertels, and C. G. Almudever, *A control microarchitecture for fault-tolerant quantum computing*, Microprocess. Microsyst. **70** (2019), 21–30.
31. L. Lao, B. van Wee, I. Ashraf, J. Van Someren, N. Khammassi, K. Bertels, and C. G. Almudever, *Mapping of lattice surgery-based quantum circuits on surface code architectures*, Quantum Sci. Technol. **4** (2019), 015005.
32. H. Daniel, N. Franco, and J. D. Simon, *Lattice surgery translation for quantum computing*, New J. Phys. **1** (2017), 013034.
33. D. Litinski and F. von Oppen, *Lattice surgery with a twist: simplifying Clifford gates of surface codes*, Quantum **2** (2018), 62.
34. C. Christopher and T. C. Earl, *Universal quantum computing with a twist-free and temporally encoded lattice surgery*, PRX Quan. **3** (2022), 010331.
35. Y. Tomita and K. M. Svore, *Low-distance surface codes under realistic quantum noise*, Phys. Rev. A **90** (2014), 062320.
36. D. Litinski, *A game of surface codes: large-scale quantum computing with lattice surgery*, Quantum. **3** (2019), 128.
37. K. S. Jin and G. I. Cha, *Multilayered logical qubits and synthesized quantum bits*, Quantum Sci. Technol. **8** (2023), 035008.
38. S. Nagayama, A. G. Fowler, D. Horsman, S. J. Devitt, and R. V. Meter, *Surface code error correction on a defective lattice*, New J. Phys. **19** (2017), 023050.
39. J. Edmonds, *Paths, trees, and flowers*, Can. J. Math. **17** (1965), 449–467.
40. V. Kolmogorov, *Blossom V: a new implementation of a minimum cost perfect matching algorithm*, Math. Program. Comput. **1** (2009), 43–67.
41. O. Higgott, *PyMatching: a Python package for decoding quantum codes with minimum-weight perfect matching*, ACM Quantum Comp. Trans. **3** (2022), no. 3, 1–16.
42. T. E. O'Brien, B. Tarasinski, and L. DiCarlo, *Density-matrix simulation of small surface codes under current and projected experimental noise*, NPJ Quantum Info. **3** (2017), 39.
43. J. H. On, C. Y. Kim, S. C. Oh, S. M. Lee, and G. I. Cha, *A multi-layered Pauli tracking architecture for lattice surgery-based logical qubits*, ETRI J. **45** (2023), 462–478.
44. E. T. Campbell, B. M. Terhal, and C. Vuillot, *Roads towards fault-tolerant universal quantum computation*, Nature **549** (2017), no. 7671, 172–179.
45. K. S. Jin and G. I. Cha, *QPlayer: lightweight, scalable, and fast quantum simulator*, ETRI J. **45** (2022), 304–317.

AUTHOR BIOGRAPHIES

Sang-Min Lee earned her BS degree in computer engineering at Inha University, Incheon, Republic of Korea in 1991. She has been with the Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea since 1991, where she has worked on the development of the SCSI and FC RAID systems, distributed parallel file systems, dual-mode big data platforms, and simulation technology for digital twins. She is currently a principal researcher. Her research interests include distributed storage, extreme storage, and quantum operating systems.

Ki-Sung Jin received his BS and MS degrees in computer engineering from Jeonbuk National University, Jeonju, Republic of Korea, in 1999 and 2001, respectively. Since 2001, he has been with the Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea, where he has worked on the development of the cluster database, distributed parallel filesystem, dual-mode big data platform, and simulation technology for digital twins. He is currently a principal researcher, and his current research interests include distributed storage, extreme storage, and quantum operating systems.

Soo-Cheol Oh earned his BS, MS, and PhD degrees in computer engineering in 1995, 1997, and 2003, respectively, from Pusan National University, Pusan, Republic of Korea. From 1997 to 1998, he worked as a research engineer at LG Multimedia Research Laboratory. Since 2005, he has been a principal researcher at the Electronics and Telecommunications Research Institute in Daejeon, Republic of Korea. His research interests include quantum computing and cloud systems.

Jin-Ho On received his MS and PhD degrees in computer engineering in 2008 and 2012, respectively, from Jeonbuk National University, Jeonju, Republic of Korea. He has been working at the Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea since 2013. He contributed to the development of energy-aware operating systems, microservice architectures for AI computing, and quantum operating systems for fault-tolerant quantum computing. He is currently a senior researcher. His current research interests include quantum computing architectures and fault-tolerant quantum operation systems.

Gyu-Il Cha received his BS and MS degrees in computer science in 1998 and 2000, respectively, at Korea University, Seoul, Republic of Korea. He has been with the Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea, since 2000 and is currently working as a principal researcher. His research interests include the development of quantum operating systems for fault-tolerant quantum computing. He is involved in the technological development of operating systems, memory virtualization, supercomputing, microservice architectures, and extreme storage systems.

How to cite this article: S.-M. Lee, K.-S. Jin, S.-C. Oh, J.-H. On, and G.-I. Cha, *Optimal execution of logical Hadamard with low-space overhead in rotated surface code*, ETRI Journal **46** (2024), 759–773, DOI 10.4218/etrij.2024-0129.