Quantum Information Processing (2025) 24:164
https://doi.org/10.1007/s11128-025-04789-4

# Architectures for lattice surgery-based surface code quantum computing under 3-dimensional nearest-neighbor connectivity

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

Received: 12 September 2023 / Accepted: 16 May 2025


**Abstract**

Efforts are being made to develop quantum processors beyond planar connectivity and quantum error correction codes suitable for 3-dimensional structures. In this study, we propose 3-dimensional architectures of logical data qubits and logical ancilla qubits based on surface codes using lattice surgery in situations where 3-dimensional nearest-neighbor connectivity between physical qubits is guaranteed. We also propose a method for performing CNOT operations on the proposed architecture. We present a physical qubit structure that stacks 2-dimensional physical qubit layers and adds several additional physical qubits for lattice surgery between adjacent layers. Our design permits the fast transversal application of CNOT operations between logical qubits in a nearest-neighbor relationship on adjacent layers, which is three times faster than the speed of standard lattice surgery CNOTs. Our design also permits the performance of CNOT operations between logical qubits that are far from each other using fast SWAP operations with transversal CNOT and lattice surgery. To demonstrate the advantages of our architecture, we present the required number of qubits and time for benchmark circuits on the proposed architecture and compare the required qubits and time on the checkerboard architecture and row-type architecture which are a 2-dimensional architectures to show the advantages of our architecture.

**Keywords** Quantum error-correcting code · Quantum architectures · Quantum computing · Transversal CNOT gate · Lattice surgery

✉ Jun Heo
junheo@korea.ac.kr

Jinyoung Ha
ksuwer@korea.ac.kr

Yujin Kang
yujin20@korea.ac.kr

Jonghyun Lee
ljh0523@korea.ac.kr

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

Published online: 05 June 2025

## 1 Introduction

A variety of quantum computing techniques, ranging from algorithms to physical devices, have been actively investigated since Shor [1] proposed a polynomial-time quantum algorithm to find discrete logarithms and factor integers. There has been a significant improvement in quantum devices in the last few years, both in terms of physical error rates and number of qubits. IBM, for instance, has made several devices with 5–433 qubits with moderate error rates accessible via the cloud [2]. Google demonstrated that it can lower the error rate of calculations by making their quantum code bigger [3].

However, current quantum computers have clear limitations, such as high gate error rates and a limited number of physical qubits. For this reason, research on noisy intermediate-scale quantum operations is being actively conducted [4], and research on performing basic quantum error-correcting (QEC) code in quantum processors is also being conducted [5]. Studies have also been conducted to present quantum architectures and demonstrate efficiency by comparing the required resources, such as physical qubits, execution time, and quantum volume [6–10], as well as quantum computing software research for efficient quantum computer operation [11–19].

Research on quantum processors beyond planar connectivity to obtain 3-dimensional connectivity is also being conducted on various qubit platforms [20–24]. In [20, 21], the authors went beyond planar connectivity using through-silicon vias (TSVs) on superconductors. In [22], the connectivity was achieved between different microchip modules in trapped ion. In [23], the connectivity between different zones in the atom array was secured. Three-dimensional quantum error correction codes based on three-dimensional (3D) connectivity have also been studied [25, 26]. In [24], their system achieved high fidelities for two-qubit gates, 3D connectivity, and fully programmable single-qubit rotations and mid-circuit readouts by employing logical-level control within a zoned architecture for reconfigurable neutral atom arrays. Other studies [27] highlight the need for 3D placement of qubits for efficient quantum computation.

In other studies, resonant cavities with transmon qubits arranged in a 2.5D architecture have been found to be efficient in implementing surface codes with significant hardware savings and enhancements in performance/fidelity [28].

The contributions of this study are as follows. First, we propose a physical qubit placement that can perform a transversal CNOT operation and lattice surgery of a rotated surface code under 3D nearest connectivity conditions. Second, we propose two ways to arrange logical data qubits and logical ancilla qubits in the proposed physical qubit placement: Type 1 and Type 2 architectures. Third, we present a logical operation method for each 3D architecture, and the advantages of 3D architecture are presented by comparing the two-dimensional placement method checkerboard architecture and row-type architecture in terms of the number of logical qubits, time required, and space–time (the product of the number of logical qubits and time required).

The remainder of this paper is organized as follows. In Sect. 2, we briefly discuss the rotated surface codes and lattice surgery technique used in this study. Thereafter, we describe the previous logical qubit layout and briefly explain the circuit mapping procedures. In Sect. 3, we propose a placement method for physical qubits and explain

its advantages. We also propose various placement methods for logical qubits and a logical operation method for placement. In Sect. 4, we compare the estimated resource use of our 3D layouts with checkerboard architecture and row-type architecture when performing several benchmark circuits. Section 5 provides the concluding statements.

## 2 Preliminary

Section 2.1 discusses the rotated surface codes and the lattice surgery technique used in this study. Section 2.2 describes the previous logical qubit layout and briefly describes circuit mapping procedures. Finally, in Sect. 2.3, we explain how logical CNOT is performed using transversal CNOT on the surface code.

### 2.1 Rotated surface code and lattice surgery

The rotated surface codes [29] are $\frac{\pi}{4}$ rotated versions of planar surface codes. $X$ and $Z$ stabilizers of rotated surface codes recognize $Z$ and $X$ defects, similar to planar surface codes. Physical $X$ or $Z$ operations along the $Z$-boundary or $X$-boundary data qubits define each logical $X$ or $Z$ operator, respectively. One bit of logical information is encoded in the code distance $d$ logical qubit, which is made up of $d^2$ data qubits and $d^2 - 1$ stabilizer qubits. Compared to planar surface code, rotated surface code uses only about half the qubits to achieve the same code distance. In addition, rotated surface code uses only about 75% of the qubits compared to planar surface code to achieve the same logical error rate [30]. Furthermore, rotated surface codes have the advantage of requiring fewer additional qubits to perform lattice surgery compared to planar surface codes. Consequently, our study is conducted under the assumption that the rotated surface code is utilized, although our proposed method is applicable to both rotated and planar surface code-based architecture.

The method for constructing a single logical qubit in three dimensions has also been investigated [25]. Their structure enables the implementation of transversal CZ gates and CCZ gates, thereby eliminating the need for magic state distillation. Despite the advantages of the 3D surface code, it requires $O(d^3)$ physical qubits to construct a logical qubit with distance $d$. This significantly increases the number of physical qubits required compared to the 2D surface code, which requires $O(d^2)$ physical qubits to create a single logical qubit of distance $d$. Therefore, our study focuses on the rotated surface code, a type of 2D surface code.

Lattice surgery is an effective technique utilized in the implementation of surface codes for quantum error correction. This method facilitates the manipulation and interaction of logical qubits that are encoded within a two-dimensional lattice of physical qubits, all while minimizing the need for significant overhead or disruption to the encoded information. The advantages of lattice surgery can be summarized as follows. First, by harnessing the topological properties of surface codes, lattice surgery ensures the fault tolerance of logical qubits throughout operations. Second, it enables the execution of complex quantum operations without requiring a substantial number of additional qubits, thus optimizing resource efficiency. Finally, lattice surgery offers

a flexible approach for the creation, merging, and splitting of logical qubits, which is essential for the scalability of quantum computing architectures.

The two primary methods of lattice surgery are merging and splitting [29]. In planar and rotated surface codes, these methods provide a logical universal quantum computation (UQC) set with only the nearest-neighbor CNOT operation. A UQC set refers to a collection of quantum gates that, when combined appropriately, are capable of approximating any unitary operation on a quantum system. By activating the stabilizers between two logical qubits, the merging process unifies two logical bits of data. $Z(X)$ boundary-merging results from the transformation of two weight stabilizers at $Z(X)$ boundaries into four weight stabilizers and the activation of $X(Z)$ stabilizers; the multiplication of the enabled stabilizer measurement outcomes is equal to measuring logical $XX(ZZ)$ for the associated logical qubits. The logical information in the merged qubit is comparable to that in the XOR operation on a $Z(X)$ basis following the $Z(X)$ boundary-merging process. In contrast, splitting disables the stabilizer in one logical qubit, resulting in the generation of two logical qubits. Disabling the $X(Z)$ stabilizers causes $Z(X)$ boundary splitting, which is similar to merging.

Figure 1 shows the process of logical CNOT using lattice surgery. The processes of merging and splitting each require $d$ cycles; consequently, the overall time needed amounts to $3d$ cycles. Note that the target qubit, intermediate qubit, and control qubit form a right angle. Due to this configuration, the checkerboard architecture is an appropriate structure for executing CNOT operations based on lattice surgery.

### 2.2 Logical qubit layout

A virtual layer called a qubit architecture is responsible for configuring logical qubits and effectively managing time and space resources. The placement of qubits supporting lattice surgical techniques in tile-based architecture (t-arch) and checkerboard architecture (c-arch) was discussed in [7]. As shown in Fig. 1, the execution of a lattice surgery-based CNOT gate requires that the control and target qubits must be positioned within patches that create a 90-degree elbow shape. In the c-arch, neighboring data patches are consistently arranged in such 90-degree configurations, facilitating the direct implementation of a lattice surgery-based CNOT gate between them. In our study, we focus on the c-arch among c-arch and t-arch because it uses fewer logical ancilla qubits, resulting in better qubit efficiency. The CNOT and SWAP qubit operations can be carried out between logical qubits that are close to each other. The ratio of logical data qubits to all logical qubits, which serves as a measure of the qubit efficiency of the c-arch, is 1/2, while the efficiency of the t-arch is 1/4.

A study on row-type architecture (r-arch), which arranges logical data qubits and logical ancilla qubits in a row, has also been published [10]. As shown in Fig. 2, logical data qubits are placed in any of the blue patches in the r-arch, while logical ancilla qubits are found in the red patches. The qubit efficiency of the r-arch is also 1/2. The r-arch has the advantage of requiring fewer logical SWAP operations because CNOT operations are possible even between logical data qubits that are far apart from each other.

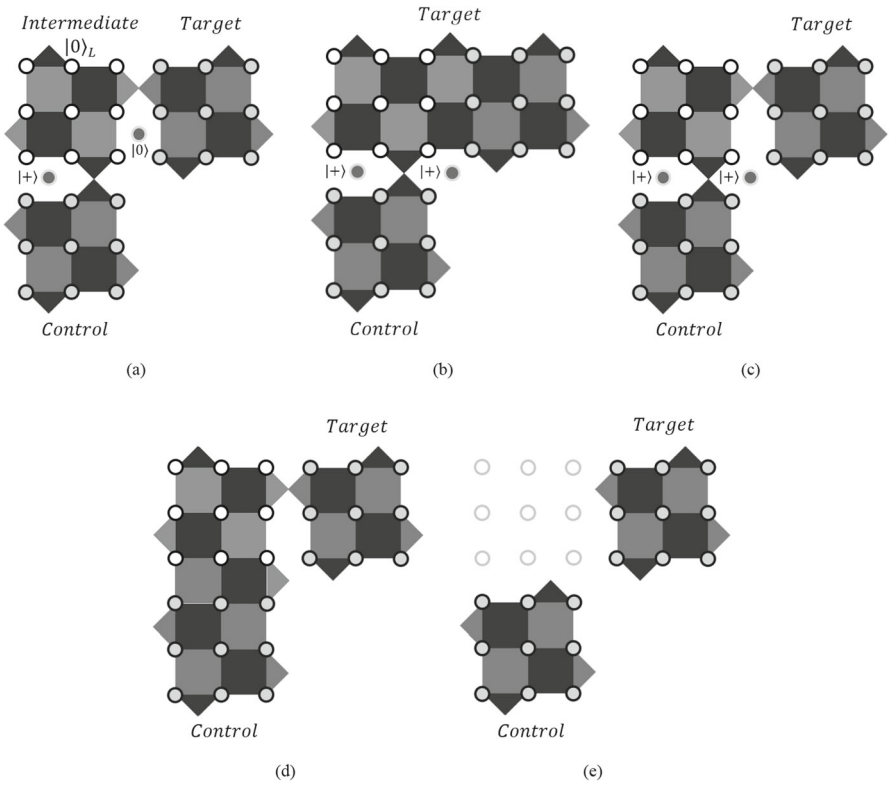


**Fig. 1** Process of performing logical CNOT using lattice surgery. The circle represent data qubits, and black (gray) faces represent $X(Z)$ stabilizers. The $X(Z)$ boundary refers to a boundary where the $X(Z)$ stabilizer exists. **a** Initial placement of control, target, and intermediate qubits. Intermediate qubit must be prepared in $|0\rangle_L$ state. **b** $Z$ boundary Merging between target qubit and intermediate qubit. **c** Splitting target qubit. **d** $X$ boundary Merging between control qubit and intermediate qubit. **e** Trimming. **c**, **d** Can be performed simultaneously

### 2.3 Transversal logical CNOT gates

In quantum computing, a transversal operation is a quantum gate or operation applied to multiple qubits that preserves the integrity of quantum error-correcting codes, especially within the framework of fault-tolerant quantum computation [31]. Transversal operations are designed to ensure that errors affecting one qubit do not propagate across the entire system, thus maintaining the integrity of the quantum information. In the case of Calderbank–Shor–Steane (CSS) code, logical CNOT can be made transversally. Because the surface code is a kind of CSS code, logical CNOT in surface code can also be made transversally. Although theoretically possible, owing to difficulties in actual placement of transversal CNOT, methods of performing CNOT through lattice surgery, deformation, etc., have been studied [29, 32]. However, studies are underway to achieve beyond planar connectivity [20–24], and transversal CNOT will also be feasible because 3D placement is also possible if planar connectivity is exceeded. A

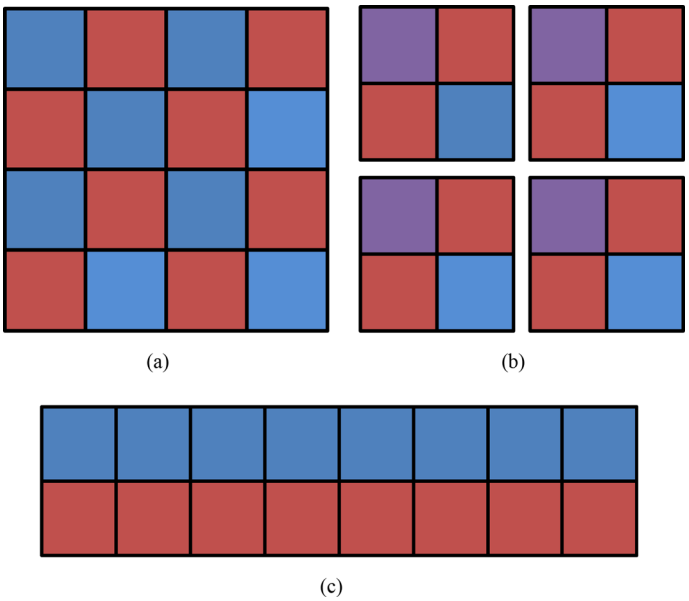


**Fig. 2** Layouts of checkerboard architecture, tile-based architecture, and row-type architecture, which are surface code-based 2D logical qubit placement methods. Checkerboard and row-type architectures are mainly studied in this paper. In **a**, **c**, the blue and red patches represent the logical data qubit and the logical ancilla qubit, respectively. **a** 8-logical data qubit checkerboard architecture layout. **b** 4-logical data qubit tile-based architecture layout. In a cluster consisting of 4 logical qubits, data qubits are assigned to one of the purple and blue patches, and ancilla qubits are assigned to the other colors. **c** 8-logical data qubit row-type architecture layout (Color figure online)

method for arranging a quantum process layer and quantum memory layer and performing an operation using transversal CNOT between the surface code logical qubits of the processor and memory layers has also been proposed [28].

## 3 Proposed qubit placements and their advantages

In this section, we propose the placement of physical and logical qubits under the condition of 3D nearest-neighbor connectivity and explain the advantages of the scheme and the operation method in that placement. Figure 3 shows the connectivity and placement of physical qubits as assumed in this study. First, we propose a physical qubit placement in which 2-dimensional (2D) qubit placements are stacked and interlayer physical qubits are placed between the stacked layers to perform lattice surgery. We show that transversal logical CNOT is possible between logical qubits in the same position of different layers and that lattice surgery using interlayer physical qubits can also be performed. Second, we propose two logical qubit placement methods based on the proposed physical qubit placement and explain the advantages and operation methods of each placements. We demonstrate that the Type 1 placement method has the advantage of making good use of transversal operations and using fewer logical

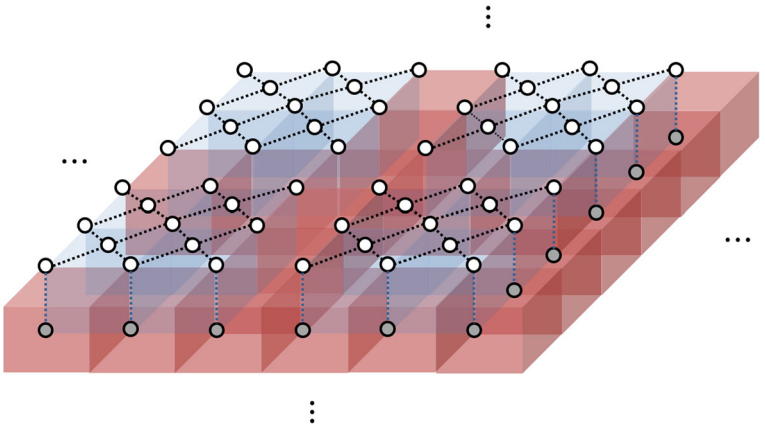

**Fig. 3** Connectivity between physical qubits assumed in this paper and physical qubit placements. The white circle represents the upper-layer physical qubit, and the gray circle represents the lower-layer physical qubit. For visibility, only a portion of the lower-layer qubits are displayed. The black dotted line indicates in-layer connectivity, and the blue dotted line indicates connectivity between layers. The interlayer physical qubit is positioned at the center of the red box and exhibits connectivity with the eight physical data qubits located at the vertices of the box (Color figure online)

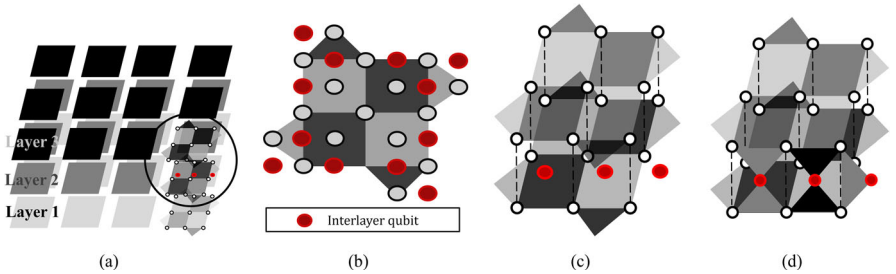


**Fig. 4** Configuration of physical qubits proposed in this study. **a** 2D layers are stacked, and interlayer physical qubits (red dots) are placed along the edges of logical qubits (black, gray, and light gray parallelograms) between layers. **b** Top view of layer 1 and interlayer. To enable lattice surgery from all sides, red-dot interlayer physical qubits surround the entire rim of the logical qubits. **c** An enlarged view of the circle in **a**. A dotted line connection between data qubits between layers means that connectivity exists between the two data qubits. Logical CNOT can be performed through transversal CNOT using this connectivity between layers. **d** Merge between logical qubits of different layers. The merge operation can be performed by turning off the stabilizer on the edge of the surface to be merged and turning on the stabilizer using the interlayer physical qubit (Color figure online)

ancilla qubits. In addition, we show that the Type 2 placement method can easily perform multiple operations simultaneously using lattice surgery and logical ancilla qubits, and the multi-target CNOT operation can also be easily performed.

### 3.1 Physical qubit placement and its advantages

We propose a 3D physical qubit placement method when the 3D nearest-neighbor connectivity is satisfied. As shown in Fig. 4, the 2D layers of lattice placement are

stacked in 3D, and the interlayer physical qubits represented by the red dot in figure 2 are placed to perform lattice surgery between different layers on the edge of the logical qubit. As shown in Fig. 4b, when the distance of the surface code is $d$, $d$ interlayer physical qubits are required for each boundary; therefore, $4d$ interlayer physical qubits are required for each logical qubit between the layers. Because the rotated surface code with distance $d$ is composed of $2d^2 - 1$ physical qubits, the additional $4d$ required for interlayer physical qubit placement can be ignored when $d$ is sufficiently large. Furthermore, when lattice surgery is not performed, interlayer physical qubits can replace the qubits used for stabilizer measurements, as represented by the triangles in Fig. 4. In Fig. 4d, when the qubit on which the lattice surgery is performed is unfolded, it has the same shape as the qubit on which the lattice surgery is performed in a 2D architecture. Therefore, through the interlayer physical qubits, lattice surgery on 3D architecture can be handled in the same manner as in 2D architecture.

Through the placement shown in Fig. 4, a logical CNOT between logical qubits with different layers but the same position can be performed as a transversal CNOT. Therefore, the CNOT operation can be performed in $d$ cycles, and the SWAP operation can be performed in $3d$ cycles [28]. This can save both qubit and time resources compared to CNOT using lattice surgery requiring $3d$ cycles and one additional logical ancilla qubit and SWAP operation requiring $9d$ cycles and one additional logical ancilla qubit.

When compared to the 3D–3D lattice surgery and 2D–3D lattice surgery discussed in [25], the approach proposed in this paper is fundamentally a more general lattice surgery method, namely 2D–2D lattice surgery. As such, it requires fewer qubits than the 3D–3D lattice surgery, and unlike the 2D–3D lattice surgery, it allows for the straightforward merging of $X$ boundaries.

### 3.2 Logical qubit placements and operations: Type 1 architecture

As shown in Fig. 5, logical ancilla qubits are placed on the top layer and logical data qubits are placed on the remaining layers. Logical ancilla qubits allow lattice surgery to be performed between qubits in different positions (different $x$- and $y$-coordinates). Because lattice surgery is not required between two different data qubit layers, interlayer physical qubits are not required, and only interlayer physical qubits must be placed between the data qubit and ancilla qubit layers. The operation between logical data qubits of different layers at the same position can be performed using a transverse CNOT when two qubits are adjacent. When two qubits are separated, the positions of the qubits are moved adjacent through a transversal SWAP operation. Subsequently, a logical CNOT operation can be performed using transverse CNOT.

If logical CNOT between qubits in different positions (different $x$- and $y$-coordinates) is to be performed, lattice surgery using logical ancilla qubits is used. The specific procedure for the CNOT between different positions is shown in Fig. 6. To merge with the logical ancilla qubit, the logical data qubits for which the CNOT operation is to be performed are placed directly below the ancilla qubit layer using transversal SWAP. The CNOT operation is then performed via lattice surgery through merging and splitting.

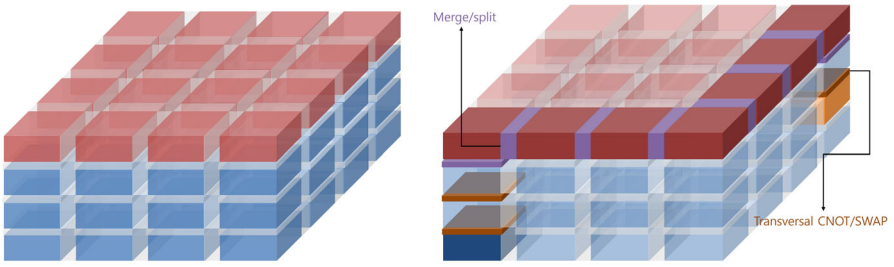


**Fig. 5** Type 1 architecture and operation methods in that placement. **a** Red and blue boxes indicate logical ancilla and logical data qubits. The operation between different positions uses transversal SWAP and lattice surgery with a logical ancilla qubit. **b** Performing CNOT or SWAP in Type 1 architecture. The dark blue box and orange box represent the control qubit and target qubit, respectively, of the CNOT operation to be performed, and the logical data qubits not directly related to the CNOT operation are shown semi-transparently. Transversal CNOT and SWAP operations are shown in brown boxes and merge and split operations are shown in purple. To perform CNOT between two logical data qubits represented by dark blue boxes, the logical data qubit is located directly below the ancilla qubit layer through transversal SWAP operation, and the CNOT operation is then performed by lattice surgery using logical ancilla qubits (dark red) (Color figure online)

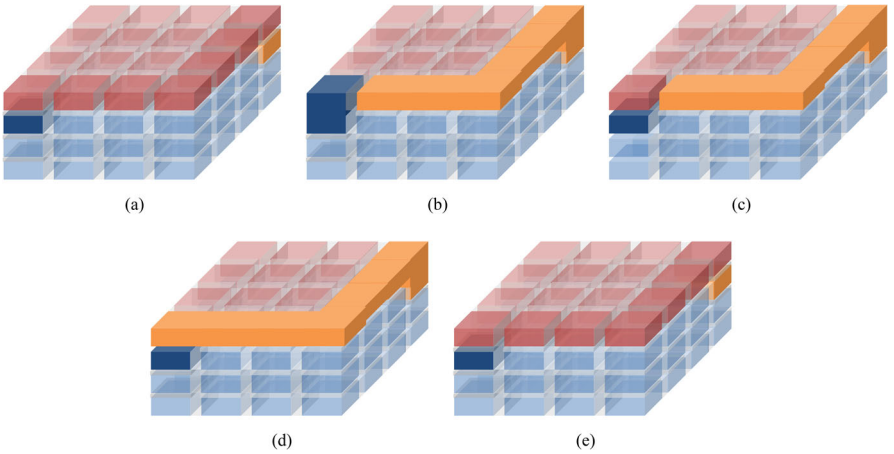


**Fig. 6** Process of CNOT operation for the control (dark blue) and target (orange) logical qubits. **a** Logical data qubits to be operated on are moved to the ancilla qubit layer and through the transversal SWAP operation. **b** A merge is performed between the blue control qubit and the logical ancilla qubit of the upper layer, and a merge is performed between the orange target qubit and the logical ancilla qubits. **c** Split is performed between the control qubit and the logical ancilla qubit. **d** Merge between the logical ancilla qubit and the merged orange qubit. **e** Shrinking the merged orange qubits and re-initializing the logical ancilla qubits. Steps **c**, **d** can be performed simultaneously (Color figure online)

When logical CNOT is performed in the above method, the feasibility of simultaneous multiple CNOT operations depends on the position of the qubits to perform the CNOT operations. In CNOT using lattice surgery, paths are created by merging logical ancilla qubits during the merging process. Therefore, when CNOTs using lattice

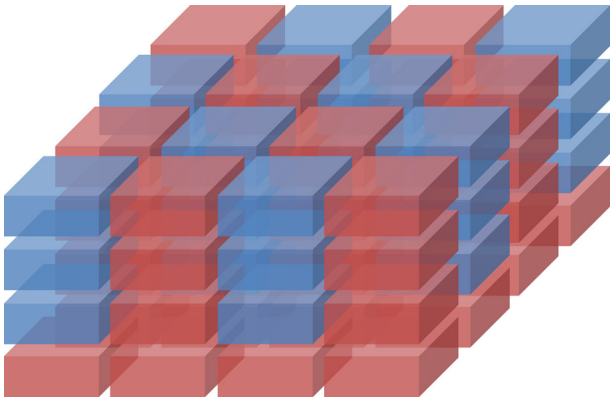

**Fig. 7** Structure of Type 2 architecture. It is composed of a logical ancilla qubit layer at the bottom and a checkerboard architecture stacked on top of it. It has the advantage that CNOT operation between arbitrary logical data qubits can be performed in $3d$ cycles using lattice surgery

surgeries are performed simultaneously if the paths overlap, they cannot be performed simultaneously.

In the case of Type 1 architecture, more transversal SWAPs are required as more layers are stacked, which leads to an extension of the operation execution time. Therefore, we propose a Type 2 architecture, a 3D architecture suitable for algorithms that use many qubits.

### 3.3 Logical qubit placements and operations: Type 2 architecture

The Type 2 architecture is composed by placing a logical ancilla qubit layer at the bottom and stacking a checkerboard architecture on top of it. Because the Type 2 architecture performs operations mainly in lattice surgery, interlayer physical qubits for lattice surgery must always exist between the ancilla qubit layers. In the Type 2 architecture, because a logical ancilla qubit exists next to an arbitrary logical data qubit, connection through lattice surgery is always possible between two logical data qubits. Therefore, even if the layer is large, a CNOT can be performed at an arbitrary position through lattice surgery; thus, an arbitrary CNOT can be performed in $3d$ cycles. In addition, because logical data qubits are surrounded by logical ancilla qubits, the accessibility between the qubits is improved, thereby increasing the number of operations that can be performed simultaneously. Figure 8 shows the process by which the two CNOTs are performed simultaneously. As shown in Fig. 8, the Type 2 architecture has the advantage of increasing the probability of performing multiple CNOT operations simultaneously because it uses many logical ancilla qubits.

Another advantage of the Type 2 architecture is that it is easy to perform multi-target CNOT operations. Herr et al. [33] showed that a multi-target CNOT operation can be performed in $3d$ cycles regardless of the number of targets when the target logical data qubits are lined up and logical ancilla qubits are lined up next to them. As shown in Fig. 7, because the Type 2 architecture has logical data qubits arranged in the form of


**Fig. 8** The process of two CNOT operations in the Type 2 architecture. **a** Dark blue represents CNOT 1 and orange represents CNOT 2. **b** In both CNOT 1 and CNOT 2, the target qubit and the control qubit are adjacent through merging with the logical ancilla qubit, therefore CNOT can be performed through lattice surgery. Because these processes can proceed simultaneously, it can be confirmed that CNOT 1 and CNOT 2 can be performed simultaneously (Color figure online)

a column and logical ancilla qubits next to it, a multi-target CNOT operation targeting several logical data qubits in one column can be performed in $3d$ cycles. The Type 2 architecture has the advantage that the computation time can be reduced compared to the Type 1 architecture, although the number of logical ancilla qubits used increases, so the total number of qubits used increases.

## 4 Results

For the Type 1 architecture (3D), Type 2 architecture (3D) and checkerboard-type architecture (2D), row-type architecture (2D), the time expressed in the surface code error syndrome measurement cycle, the number of logical qubits, and the space–time of benchmark circuits are compared. Space-time is the product of time and the number of logical qubits. We used [[7, 1, 3]] code execution circuit, [[15, 1, 3]] as benchmark circuits. The initial placement of the circuit was arranged sequentially without optimization. The scheduling of logical operations was performed manually without optimization.

Figure 9 shows the [[7, 1, 3]] circuit, and the initial placement of each architecture that performs the circuit. The result of the [[7, 1, 3]] circuit operation scheduling for the Type 1 architecture is shown in Fig. 10. It shows that $34d$ error correction cycles are required. Figure 11 shows the time and number of logical qubits used for each architecture.

In Fig. 11, qubit is the number of logical qubits consumed by the algorithm, time is the surface code error syndrome measurement cycle, and space–time is expressed as the product of time and qubit. Overall, the Type 1 architecture requires fewer logical ancilla qubits than the other architectures; therefore, the number of logical qubits used is small. Despite the low number of logical qubits used, the times for Type 1 architecture are comparable to the r-arch and smaller than the c-arch for all benchmarks. The reason why the Type 1 architecture takes a similar amount of time as the row-type architecture is due to the process involved in the Type 1 architecture, specifically, the

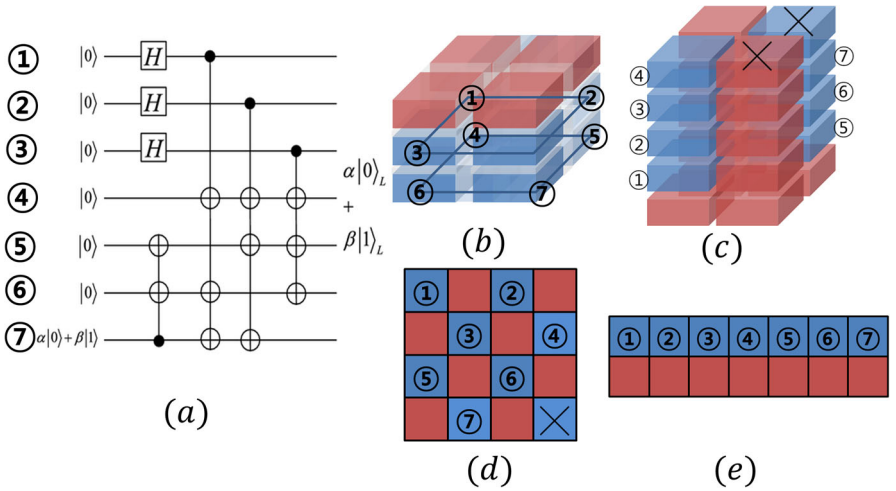


**Fig. 9** Initial placement of the [[7, 1, 3]] circuit and each architecture. The mark 'X' in **c** indicates logical qubits that are not used. **a** [[7, 1, 3]] circuit. Initial placement of **b** Type 1 architecture, **c** Type 2 architecture, **d** checkerboard architecture, and **e** row-type architecture

time it requires to elevate the logical data qubit from the lowest layer to the top using a transversal SWAP operation. In contrast to the Type 1 architecture, the Type 2 architecture requires the least time while having the largest number of logical qubits for all benchmark circuits. Therefore, the Type 1 architecture can be useful when qubits are a critical resource, and the Type 2 architecture is advantageous when a small amount of time is important. In the case of space–time, which can examine the overall required resources, it can be confirmed that for the [[7, 1, 3]] benchmark circuit, the Type 1 architecture consumes approximately 47% of the resources compared to the checkerboard architecture and approximately 83% of the resources compared to the row-type architecture. For the [[15, 1, 3]] benchmark circuit, the Type 1 architecture consumes approximately 35% of the resources compared to the checkerboard architecture and approximately 62% of the resources compared to the row-type architecture. Therefore, the [[15, 1, 3]] circuit is more advantageous for 3D architecture than the [[7, 1, 3]] circuit. The reason for this is that the larger the size of the circuit, the more operations that are performed simultaneously, the higher the saving rate of the ancilla qubits, and the higher the probability of utilizing the advantages of the 3D architecture. For both benchmark circuits, the Type 2 architecture had a slightly smaller space–time than the Type 1 architecture. Therefore, in terms of space–time, the Type 2 architecture is the most efficient among the four architectures. Because [[7, 1, 3]] circuit and [[15, 1, 3]] circuit are the main components of magic state distillation, which requires considerable resources when performing quantum circuits, being able to perform them efficiently means that many quantum circuits can be performed efficiently.

| Time | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 0 | Init | Init | Init | Init | Init | Init | Init |
| 1 | | | | | | | |
| 2 | H | H | H | | | | SWAP(7) |
| 3 | | | | | | | |
| 4 | | | | | | | |
| 5 | | | | | | | |
| 6 | | SWAP(5,2) | SWAP(6,3) | | SWAP(5,2) | SWAP(6,3) | |
| 7 | | | | | | | |
| 8 | | | | | | | |
| 9 | | | | | | CNOT(7,6) | |
| 10 | | | | | | | |
| 11 | | | | | | | |
| 12 | | | | | CNOT(7,5) | | CNOT(7,5) |
| 13 | | | | | | | |
| 14 | CNOT(1,4) | | | CNOT(1,4) | | | |
| 15 | | | | | | | |
| 16 | CNOT(1,6) | | | | | CNOT(1,6) | |
| 17 | | | | | | | |
| 18 | | | | | | | |
| 19 | CNOT(1,7) | SWAP(2,5) | | | SWAP(2,5) | | CNOT(1,7) |
| 20 | | | | | | | |
| 21 | | | | | | | |
| 22 | SWAP(4,1) | CNOT(2,7) | | SWAP(4,1) | | | CNOT(2,7) |
| 23 | | | | | | | |
| 24 | | | | | | | |
| 25 | | CNOT(2,4) | SWAP(3,6) | CNOT(2,4) | | SWAP(3,6) | |
| 26 | | | | | | | |
| 27 | | CNOT(2,5) | CNOT(3,6) | | CNOT(2,5) | CNOT(3,6) | |
| 28 | | | | | | | |
| 29 | | SWAP(5,2) | CNOT(3,4) | | SWAP(5,2) | | |
| 30 | | | | | | | |
| 31 | | | | | | | |
| 32 | | | CNOT(3,5) | | CNOT(3,5) | | |
| 33 | | | | | | | |

**Fig. 10** Result of [[7, 1, 3]] circuit operation scheduling for Type 1 architecture. 'Time' means surface code error syndrome measurement cycle. 'Init' means initialization of logical qubits

## 5 Conclusion

We proposed a 3D physical qubit placement method based on 3D nearest-neighbor connectivity. Using the presented physical qubit placement method, we propose two logical qubit placement methods and a method for performing logical operations in these placements. They reduced the number of logical qubits or the computation time by using the fact that a transversal CNOT operation between adjacent surface codes is possible, and the accessibility between logical qubits increases in the case of a 3D structure. Because the 2D architecture uses lattice surgery to perform CNOT, it takes $3d$ cycles to perform CNOT and $9d$ cycles to perform the SWAP operation, whereas the 3D architecture performs transversal CNOT between adjacent logical qubits in $d$ cycles and transversal SWAP in $3d$ cycles. In addition, when using a 3D architecture with more ancilla qubits, CNOT can be performed in only $3d$ cycles by using lattice surgery without using the SWAP operation for logical qubits at arbitrary positions.

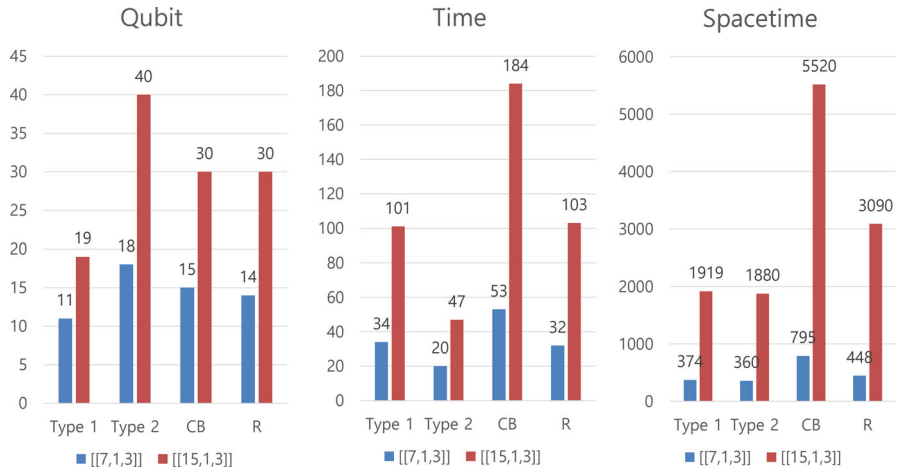


**Fig. 11** Time, logical qubits used, and space–time of benchmark circuits for each architecture. Space–time is expressed as the product of time and logical qubits used. *CB* checkerboard architecture, *R* row-type architecture

Furthermore, in a 3D architecture, depending on the location of the operation, multi-target CNOT can be performed in only $3d$ cycles.

We also compared the resources required for 3D architecture and 2D architecture by manually performing scheduling on two benchmark circuits. The Type 1 architecture maintained similar times while using fewer qubits compared with r-arch in both benchmark circuits. Although the Type 2 architecture uses more qubits, it significantly reduces the time compared to r-arch in both benchmark circuits. Regarding the space-time representing the comprehensive required resources, both 3D architectures showed some gain compared to 2D in space–time. In addition, the space–time gain of the 3D architecture increased as the scale of the benchmark circuit increases, which is expected to be universally applicable when the scale of the quantum circuit increases. Because the two 3D architectures have similar space–times despite the very different resources required in the qubit and time, it is expected that the two architectures can be compatibly selected depending on the quantum computing environment.

**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 Quantum Technology R&D Leading Program(Quantum Computing(RS-2024-00431853) through the National Research Foundation of Korea(NRF) funded by the Korean government (Ministry of Science and ICT(MSIT)). This work was supported by the IITP(Institute of Information & Coummunications Technology Planning & Evaluation)-ITRC(Information Technology Research Center) grant funded by the Korea government(Ministry of Science and ICT)(RS-2021-II211810).

**Data Availability** The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest.

## References

1. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science, pp. 124–134. IEEE (1994)
2. IBM Quantum (2023). https://quantum-computing.ibm.com/
3. Google Quantum AI: Suppressing quantum errors by scaling a surface code logical qubit. Nature **614**(7949), 676–681 (2023)
4. Preskill, J.: Quantum computing in the NISQ era and beyond. Quantum **2**, 79 (2018)
5. Satzinger, K., Liu, Y.-J., Smith, A., Knapp, C., Newman, M., Jones, C., Chen, Z., Quintana, C., Mi, X., Dunsworth, A., et al.: Realizing topologically ordered states on a quantum processor. Science **374**(6572), 1237–1241 (2021)
6. Fowler, A.G., Gidney, C.: Low Overhead Quantum Computation Using Lattice Surgery. arXiv preprint arXiv:1808.06709 (2018)
7. 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)
8. Litinski, D.: A game of surface codes: large-scale quantum computing with lattice surgery. Quantum **3**, 128 (2019)
9. Chamberland, C., Campbell, E.T.: Universal quantum computing with twist-free and temporally encoded lattice surgery. PRX Quantum **3**(1), 010331 (2022)
10. 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)
11. Abhari, A.J., et al.: Scaffold: Quantum programming language. Princeton University, New Jersey (2012)
12. JavadiAbhari, A., Patil, S., Kudrow, D., Heckey, J., Lvov, A., Chong, F.T., Martonosi, M.: ScaffCC: scalable compilation and analysis of quantum programs. Parallel Comput. **45**, 2–17 (2015)
13. Steiger, D.S., Häner, T., Troyer, M.: ProjectQ: an open source software framework for quantum computing. Quantum **2**, 49 (2018)
14. LaRose, R.: Overview and comparison of gate level quantum software platforms. Quantum **3**, 130 (2019)
15. Shi, Y., Tao, R., Li, X., Javadi-Abhari, A., Cross, A.W., Chong, F.T., Gu, R.: CertiQ: A Mostly-Automated Verification of a Realistic Quantum Compiler. arXiv preprint arXiv:1908.08963 (2019)
16. Smith, R.S., Peterson, E.C., Skilbeck, M.G., Davis, E.J.: An open-source, industrial-strength optimizing compiler for quantum programs. Quantum Sci. Technol. **5**(4), 044001 (2020)
17. McCaskey, A.J., Lyakh, D.I., Dumitrescu, E.F., Powers, S.S., Humble, T.S.: XACC: a system-level software infrastructure for heterogeneous quantum-classical computing. Quantum Sci. Technol. **5**(2), 024002 (2020)
18. Sivarajah, S., Dilkes, S., Cowtan, A., Simmons, W., Edgington, A., Duncan, R.: t|ket⟩: a retargetable compiler for NISQ devices. Quantum Sci. Technol. **6**(1), 014003 (2020)
19. Khammassi, N., Ashraf, I., Someren, J., Nane, R., Krol, A., Rol, M.A., Lao, L., Bertels, K., Almudever, C.G.: OpenQL: a portable quantum programming framework for quantum accelerators. ACM J. Emerg. Technol. Comput. Syst. (JETC) **18**(1), 1–24 (2021)
20. Rosenberg, D., Kim, D., Das, R., Yost, D., Gustavsson, S., Hover, D., Krantz, P., Melville, A., Racz, L., Samach, G., et al.: 3D integrated superconducting qubits. Npj Quantum Inf. **3**(1), 42 (2017)

21. Mallek, J.L., Yost, D.-R.W., Rosenberg, D., Yoder, J.L., Calusine, G., Cook, M., Das, R., Day, A., Golden, E., Kim, D.K., et al.: Fabrication of Superconducting Through-Silicon Vias. arXiv preprint arXiv:2103.08536 (2021)
22. Akhtar, M., Bonus, F., Lebrun-Gallagher, F., Johnson, N., Siegele-Brown, M., Hong, S., Hile, S., Kulmiya, S., Weidt, S., Hensinger, W.: A high-fidelity quantum matter-link between ion-trap microchip modules. Nat. Commun. **14**(1), 531 (2023)
23. Bluvstein, D., Levine, H., Semeghini, G., Wang, T.T., Ebadi, S., Kalinowski, M., Keesling, A., Maskara, N., Pichler, H., Greiner, M., et al.: A quantum processor based on coherent transport of entangled atom arrays. Nature **604**(7906), 451–456 (2022)
24. Bluvstein, D., Evered, S.J., Geim, A.A., Li, S.H., Zhou, H., Manovitz, T., Ebadi, S., Cain, M., Kalinowski, M., Hangleiter, D., et al.: Logical quantum processor based on reconfigurable atom arrays. Nature **626**(7997), 58–65 (2024)
25. Vasmer, M., Browne, D.E.: Three-dimensional surface codes: transversal gates and fault-tolerant architectures. Phys. Rev. A **100**(1), 012312 (2019)
26. Kubica, A., Vasmer, M.: Single-shot quantum error correction with the three-dimensional subsystem toric code. Nat. Commun. **13**(1), 6272 (2022)
27. Mamgain, A., et al.: A review of developments in superconducting quantum processors. J. Indian Inst. Sci. **103**(2), 633–669 (2023)
28. Duckering, C., Baker, J.M., Schuster, D.I., Chong, F.T.: Virtualized logical qubits: a 2.5D architecture for error-corrected quantum computing. In: 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 173–185. IEEE (2020)
29. Horsman, C., Fowler, A.G., Devitt, S., Van Meter, R.: Surface code quantum computing by lattice surgery. New J. Phys. **14**(12), 123011 (2012)
30. O'Rourke, A.R., Devitt, S.: Compare the Pair: Rotated vs. Unrotated Surface Codes at Equal Logical Error Rates (2024). https://arxiv.org/abs/2409.14765
31. Nielsen, M.A., Chuang, I.L.: Quantum computation and quantum information. Cambridge University Press (2010)
32. 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)
33. Herr, D., Nori, F., Devitt, S.J.: Lattice surgery translation for quantum computation. New J. Phys. **19**(1), 013034 (2017)