Received: 4 February 2022 | Revised: 4 June 2022 | Accepted: 23 June 2022

DOI: 10.4218/etrij.2022-0037

ORIGINAL ARTICLE

ETRI Journal WILEY

A multilayered Pauli tracking architecture for lattice surgery-based logical qubits

Jin-Ho On | **Chei-Yol Kim** | **Soo-Cheol Oh** | **Sang-Min Lee** | **Gyu-Il Cha**

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

Correspondence
Jin-Ho On, Future Computing Research Division, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea.
Email: onjinho@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

In quantum computing, the use of Pauli frames through software traces of classical computers improves computation efficiency. In previous studies, error correction and Pauli operation tracking have been performed simultaneously using integrated Pauli frames in the physical layer. In such a complex processing structure, the number of simultaneous operations processed in the physical layer exponentially increases as the distance of the surface code encoding logical qubit increases. This study proposes a Pauli frame management architecture partitioned into two layers for a lattice surgery-based surface code and describes its structure and operation rules. To evaluate the effectiveness of our method, we generated a random circuit according to the gate ratios constituting the commonly known quantum circuits and compared the generated circuit with the existing Pauli frame and our method. Simulations show a decrease of about 5% over traditional methods. In the case of experiments that only increase the code distance of the logical qubit, it can be seen that the effect of reducing the physical operation through the logical Pauli frame becomes more important.

KEYWORDS
lattice surgery-based surface code, Pauli frame processing method, quantum computing

1 | INTRODUCTION

Quantum computing is a new technology that uses qubit operations based on quantum phenomena to solve complex problems in classical computers [1, 2]. During quantum computing, physical qubits interact with their surroundings. In this process, they lose their stored information, or their entanglement with other qubits disappears over time [3]. Owing to this problem, qubits struggle to maintain their state, and quantum information processing fails with any errors. Since the coherence time in superconducting qubits used for quantum computation without errors is extremely short (μs–ms), a method for increasing the computation time using an error correction technique is essential [4].

Many studies [3, 5, 6] to increase the reliability of qubits with high error rates have proposed various *Quantum Error Correction* (QEC) techniques. These QEC techniques construct a repetition code that can discriminate the error of a qubit using quantum entanglement

and perform error correction by measuring the value of the *stabilizer* in it. Thus, these approaches provide fundamental logical qubits with lower error rates [7–9].

The *surface code* [10–12] has been proposed as the most effective method for generating QEC-based logical qubits. This method provides various ways for encoding logical qubits on a two-dimensional array and for efficient computation. However, the surface code incurs significant overhead when encoding physical qubits into logical qubits. The number of available operators decreases as physical qubit operations are converted into logical qubit operations. Therefore, converting all quantum circuit operations into limited operations adds to the overhead [13]. Due to these issues, the number of physical qubit operations that the control system must execute should be minimized as much as possible by reducing the number of generated logical qubits [14].

All *Pauli gates* can be deferred during execution unless confronted with a non-*Clifford* gate, and digital computers can quickly trace them back. Considering these characteristics of quantum operations, *Pauli frames* [15–18] have been proposed to effectively process the Pauli operation in classical computers. Pauli frames can be scheduled to process *Error Syndrome Measurement* (ESM) and *decoding* for error correction in parallel. Furthermore, it can be implemented using a few physical gates.

This study proposes an efficient Pauli frame management architecture targeting the *lattice surgery-based surface code* [19]. We divide the Pauli frame management architecture into two logical and physical execution layers for efficient tracking of Pauli operations. We confirmed that the proposed method is optimized by approximately 5% compared to existing Pauli frames.

This paper is organized, as follows: Section 2 describes the relevant research on this study. Section 3 defines the problem addressed in this study, and Section 4 describes the proposed method for solving the problem. Section 5 verifies the validity of the proposed method, and finally, Section 6 concludes this study.

2 | RELATED WORKS

Physical qubits lose their state quickly, making it difficult to maintain a quantum state for an extended period. As a result, operations do not work correctly on physical qubits. The QEC method was introduced to overcome these limitations and perform quantum computation with high accuracy. QEC can encode multiple physical qubits into one logical qubit with the entanglement state through the logical qubit generation process. Logical qubits generated in this way have a lower error rate and a significantly longer coherence time than physical qubits. First, we study the *surface code*, the most efficient way to construct logical qubits.

2.1 | Surface code

The implementation method and performance of the surface code have been extensively investigated in several papers [20–24], and most quantum qubit models have adopted this approach with the advantage of relatively easy implementation of logical qubits.

Surface code-17 comprises nine data qubits and eight auxiliary qubits acting as stabilizers, as shown in Figure 1. All stabilizers are divided into *X*-stabilizers that detect a *Z* error (phase flip) of qubits and *Z*-stabilizers that detect an *X* error (bit flip) of qubits. This error detection process is known as ESM. This *syndrome measurement* is repeated as many times as the *code distance d* constituting the surface code immediately after the logical gate operation. Finally, the *error decoding* process is performed to determine the physical qubit in which the state-flip error has occurred, or the measurement error has occurred, by analyzing the measured syndromes.

All error corrections in the physical layer can be traced and processed in software through a digital computer by using a Pauli frame management architecture.

2.2 | Pauli frames

Pauli and *Clifford gates* can be processed using Pauli frames in digital computers. Therefore, a physical Pauli gate can be processed by software tracking instead of an actual physics operation using Pauli frames [15, 16, 25].

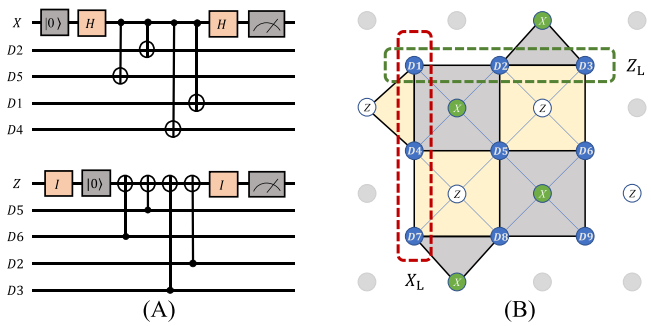

FIGURE 1 (A) shows that the *X*/*Z* stabilizer consists of a state-tracking stabilizer circuit for *X*/*Z* operation. As shown in (B), a surface code can be constructed by repeatedly configuring these stabilizers. In the surface code, logical Pauli operations X_L, Z_L and $X_L Z_L$ (Y_L) are performed through a Pauli operation consisting of a chain of data qubits.

The recording of Pauli operations for all physical qubits forms Pauli frames, which are composed of *Pauli groups*, such as $\{I, X, Z, Y\}$, depending on their mathematical properties [17, 18]. Accordingly, all Pauli operations in a quantum circuit can be calculated and stored in a digital computer using one of the $\{I, X, Z, Y\}$ operators. In Pauli frames, Pauli and Clifford operations have a special relationship that can be described as a mathematical expression, as follows:

If P represents an n-qubit Pauli group composed of the Pauli operations X, Z, and Y, the Clifford group, the normalizer of the Pauli group P, is defined as $C_n = \{V \in U_n^2 | VP_nV^\dagger = P_n\}$ for the unitary group U. Therefore, the element of the Clifford and Pauli group holds $VP|\psi\rangle = P'V|\psi\rangle$. This process has a computational complexity of $O(n^2)$, which can be processed quickly in a digital computer [26].

The advantages of Pauli frames for logical qubits described in existing studies [18, 25] based on this processing architecture are as follows. First, storing physical gate operations directly in Pauli frames reduces the number of physics gates to be executed. Similarly, logical Pauli gate operations can be stored directly in Pauli frames, eliminating the need to execute physical gates. In addition, increasing the code distance of the surface code maximizes the effectiveness of this execution. Second, because digital computers handle error correction after decoding in the ESM process, there is no need to use Pauli frames while performing ESM; thus, operation execution and decoding/error correction can be performed in parallel [25]. However, all tasks within Pauli frames must be recognized simultaneously to effectively update the Pauli record at runtime, which can result in high overhead in a large-scale physical qubit system. Although the structure of Pauli frames and their processing rules have been introduced in several previous studies [5, 16, 17], most have only described the theoretical principle [25]. Another paper [27], described from an implementation viewpoint, introduced a fault-tolerant microarchitecture, including Pauli frames, but did not describe the processing logic of Pauli frames.

3 | PROBLEM ANALYSIS

Pauli frames have many advantages for quantum computing implementation. However, the following additional considerations must be taken into account when implementing logical qubits:

- When Pauli frames exist only in the physical layer, one logical operation is expressed as physical operations equal to the code distance of the surface code. An increase in the code distance to reduce the error rate dramatically increases the number of physical Pauli frames to be simultaneously processed. We can resolve this issue by introducing the concept of Pauli frames in the logical layer.
- The lattice surgery-based logical operation process first performs computations using the Joint-Measurement action and finally performs appropriate logical Pauli operations to complete this logical operation. Every logical operation includes at least one logical Pauli operation, which can be effectively tracked and processed using the logical layer's Pauli frames. A new system-level Pauli tracking technique with a different structure and operating rules than the existing physical layer-oriented Pauli frames is required to meet this requirement.
- Because the logical layer can process all logical Pauli operations, the physical layer operates only via physical-level Pauli frames for error correction. A multilayered structure with divided roles enables the implementation of a system structure that can effectively handle error decoding.

4 | MULTILAYERED PAULI TRACKING ARCHITECTURE

A logical qubit is encoded as one patch of the surface code with an error correction function in the lattice surgery-based surface code. The X_L and Z_L operators of the logical qubit exist in the top-to-bottom and left-to-right direction, respectively, and are defined as a chain of physical qubit operations. Interaction with other patches is realized by adding new syndrome qubits to the borderline or turning added syndromes on and off [19].

As in the example of the logical *MOVE* operation in Figure 2, operations in the lattice surgery-based surface code comprise a Merge/Split operation, including Joint-Measurement and Pauli operations, to correct the logical qubit state according to the measurement result.

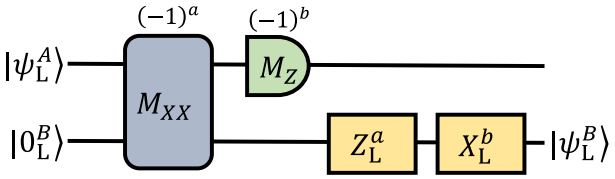

FIGURE 2 The lattice surgery-based *MOVE* operation uses one auxiliary logical qubit and consists of the M_{XX} operation, measurement operation of source logical qubit, and Pauli gates that correct the state of a logical qubit according to the measurement result.

A Pauli gate's Pauli operation can be treated as a Pauli frame when it encounters a Clifford gate, allowing it to be deferred after the Clifford operation. These principles also apply to logical Pauli operations so that they, along with Pauli operations in logical Clifford gates, can be quickly traced from the logical layer using a digital computer.

A general-purpose quantum computer has Clifford and non-Clifford gates due to the characteristics of quantum computing [28]. The lattice surgery method can configure logical operators more simply than other methods for implementing such general-purpose logical operators.

Table 1 is a brief description of gate operations in the lattice surgery method. According to the description, a logical operation can be performed in a limited form within a minimal structure compared to other implementation methods of the surface code.

We suppose that logical-level Pauli frames are applied to the logical layer, responsible for efficient execution of logical Pauli operations. Then, these physical-level Pauli frames are separated to perform error correction only during the ESM of each logical qubit. This approach reduces system complexity by distributing Pauli frames through two layers and maximizes the system efficiency by reducing unnecessary task processing.

4.1 | Overall structure

To effectively explain the multilayered Pauli frame tracking architecture proposed in this study, Figure 3 schematically depicts the simplified system block structure.

The primary processing structure of the proposed architecture is as follows. When the execution of a quantum circuit is requested from the outside, (a) the *Logical Qubit Executor* executes a quantum circuit comprising logical qubits using a lattice surgery-based operation method. (b) The *Logical Pauli Arbiter* first identifies the execution type of requested logical operation and determines its processing flow. When logical Pauli frame tracking is required, it requests processing from the *Logical Pauli Frame Unit*. If physical execution is required, it performs the actual execution using the *Physical Qubit Executor*. (c) The *QEC Cycle Generator* generates a physical qubit circuit for the ESM requested by the *Logical Qubit Executor* and the *Logical Pauli Arbiter*. It, then, requests ESM handling to the *Physical Qubit Execution Layer*. (d) The *Physical Qubit Executor* converts the requested logical operation into a series of physical operations and requests their execution to the *Physical Pauli Arbiter*. (e) The *Physical Pauli Arbiter* identifies the type of requested physical operation. It requests its execution to the *Physical Pauli Frame Unit* or the *Physical Qubit Execution Layer* for actual execution. (f) The *QEC Decoder* determines the error using the measured value of the ESM and generates a physical Pauli operation for error correction.

(g) The *Physical Pauli Frame Unit* is responsible for tracking physical Pauli operations. When the upper layer instructs a measurement operation of a physical qubit, the actual measurement result and Pauli frame are combined, and the result value is transferred to the upper layer. (h) The *P/L Measurement Translator* converts the result obtained in the *Physical Pauli Frame Unit* into the logical layer's measurement result. (i) The *Logical Pauli Frame Unit* tracks the logical Pauli operations, and the return value of the *P/L Measurement Translator* is returned as a measurement result of logical qubits to the *Logical Qubit Executor* by applying logical Pauli frames.

We detail each layer's Pauli frame handling mechanism in the following sections.

TABLE 1 Logical gate processing methods

Type	Gates	Approach
Pauli	I_L, X_L, Z_L, Y_L	All Pauli gates are traced to logical Pauli frames.
Clifford	H_L $CNOT_L$	After Joint-Measurement, a logical Pauli operation is performed according to the measurement value. ESM occurs during Merge/Split.
	S_L	Prepare $\|Y_L\rangle$ state through state injection and Magic State Distillation. Then, apply a Pauli gate based on measurement results obtained during $CNOT_L$.
Non-Clifford	T_L	Prepare $\|A_L\rangle$ state through state injection and Magic State Distillation. Then, apply an S_L gate based on measurement results obtained during $CNOT_L$.

ESM, *Error Syndrome Measurement*.

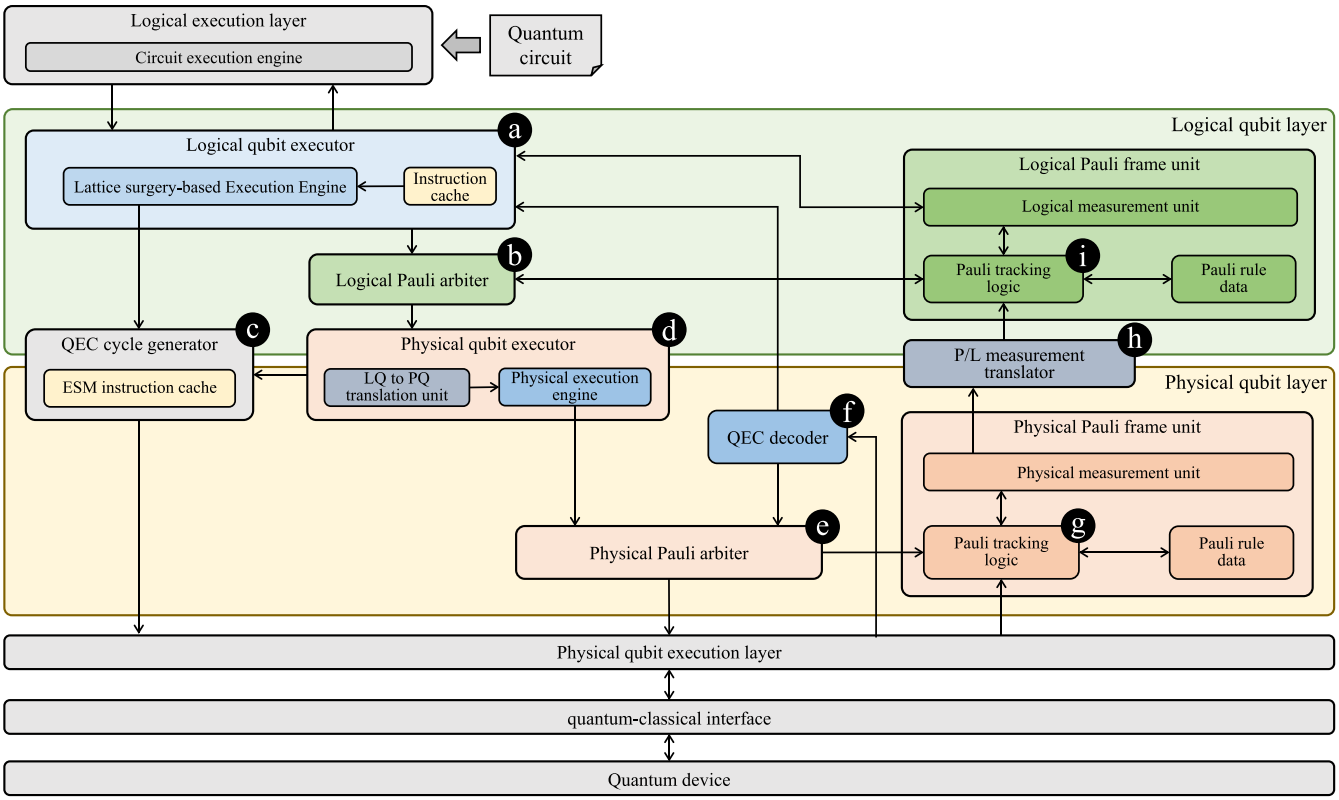

FIGURE 3 The structure of the proposed multi-layered Pauli tracking architecture and the processing flow between each module

4.2 | Logical Pauli Frame Unit

The *Logical Pauli Frame Unit* processes all logical Pauli operations in the logical layer. All logical operations of the lattice surgery technique, including logical Pauli operations, were processed using the proposed method and are stored as Pauli frames.

The logical operation of the lattice surgery technique comprises a combination of the Joint-Measurement/Measurement operations and Pauli operations reflecting the results, as shown in the example of Figure 2, and its structure is generally divided into *OPBODY* and *POSTP*.

For example, suppose the logical Clifford operation is separated into *OPBODY* and *POSTP*, as shown in Figure 4A. Considering the theoretical operating principle of Pauli frames, we can describe how to construct a logical Pauli frame, as follows:

1. The logical Pauli operation is deferred after the logical Clifford operation.
2. The delayed logical Pauli operation transforms into a new logical Pauli operation (that is, Pauli frame) by combining it with the *POSTP* in the logical Clifford operation. Then, the newly generated logical Pauli operation is stored in a logical-level Pauli frame structure.

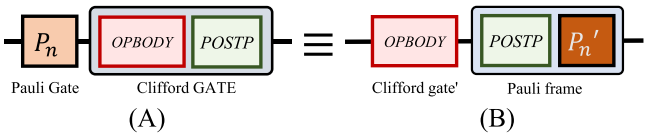

FIGURE 4 Tracking the Pauli frame on the lattice surgery-based logical operation. (A) The Pauli gate and Clifford gate representation and (B) Clifford gate and Pauli frame representation have an equal relationship by Heisenberg notation.

For example, suppose the Pauli operation P_n and Clifford operation C are successively applied to the state of the logical qubit $|\psi_L\rangle$. This is expressed as $CP_n|\psi_L\rangle$ and as $C(OPBODY, POSTP)P_n|\psi_L\rangle$ again. Owing to the mathematical characteristics of Pauli and Clifford gates, the Pauli gate can move behind the Clifford gate so that we can define the state after this rule as $P_n'POSTPC(OPBODY)|\psi_L\rangle$.

Table 2 shows the basic rules in Pauli frames, which describe the relationship between the Clifford and Pauli gates through Heisenberg notation. In general, the Pauli tracking architecture can use these rules to construct Pauli frames. Suppose we apply these basic rules to the internal processing rules of a lattice surgery-based logical operation, as shown in Figure 5. Then, the performance

of effective operations on logical qubits can also be predicted.

In the circuit of Figure 5, if the state of control and target input qubits of the logical *CNOT* gate is $X_L \bigotimes I_L$, it can be transformed to $X_L \bigotimes X_L$ after Clifford by the Heisenberg notation. After transforming, the $X_L \bigotimes X_L$ can be integrated with the logical Pauli operation inside the logical Clifford gate.

More specifically, the state of control and target output qubits of the logical *CNOT* gate is determined according to the result of the Joint-Measurement/Measurement operation in the following way. When the input state of the $CNOT_L$ gate is $X_L \bigotimes I_L$, the Pauli frame of the control qubit is obtained by combining ${Z_L}^{a+c}$ with the previous Pauli frame P_n' (where, X_L). M_{XX}, M_{ZZ}, and M_X determine ${Z_L}^{a+c}$. The Pauli frame of the target qubit is calculated as $X_L {X_L}^b$ by combining ${X_L}^b$, which M_{ZZ} determines, with the previous Pauli frame P_m' (where, X_L), which means the same as ${X_L}^{b+1}$.

Table 3 shows the final Pauli frame handling rules for logical *CNOT* gates determined in the same way above.

The *Logical Pauli Frame Unit* finally stores the Pauli frame determined according to the rules in Table 3 after reflecting the Joint-Measurement/Measurement operation. In Appendix A, we separately describe Pauli frame handling rules of logical operations other than the $CNOT_L$ gate.

TABLE 2 Rules for applying Pauli frames to Pauli and Clifford gates

Gates	Rules
H	$X \rightarrow Z$ $Z \rightarrow X$
S	$X \rightarrow Y$ $Z \rightarrow Z$
CNOT	$X \bigotimes I \rightarrow X \bigotimes X$ $I \bigotimes X \rightarrow I \bigotimes X$ $Z \bigotimes I \rightarrow Z \bigotimes I$ $I \bigotimes Z \rightarrow Z \bigotimes Z$

As shown in Figure 6, all operations in the *Logical Qubit Layer* are performed according to the following procedure:

1. When the execution of a new logical operation starts, the *Logical Qubit Executor* identifies the operator type and generates a lattice surgery-based logical operation circuit separated into two structures: *OPBODY* and *POSTP*.
2. The *Logic Pauli Arbiter* determines the following execution flow for the generated logical operation circuit according to the Pauli frame processing rules in Table 4:

- If an *OPBODY* needs to be processed, the *Logical Pauli Arbiter* requests its execution to the *Physical Qubit Layer*.
- If a *POSTP* is included in the logical Clifford operation, the *Pauli Tracking Logic* handles the Pauli frame processing operation. The previous Pauli frame, stored in Pauli tracking data, and *POSTP* operations are combined, generating a new Pauli frame.

Under the above conditions, the *Logical Pauli Arbiter* delegates the subsequent execution to the appropriate processing units.

4.3 | Physical Pauli Frame Unit

The *Physical Qubit Layer* converts commands transferred from the *Logical Qubit Layer* into physical qubit control commands, generates ESM codes for periodic error checking execution, and requests their execution in physical hardware.

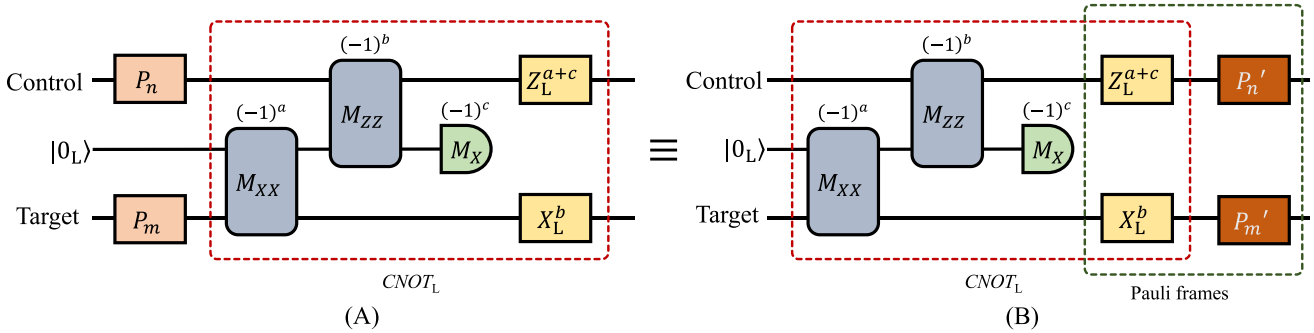

FIGURE 5 The lattice surgery-based $CNOT_L$ operation with three logical qubits, including auxiliary qubits. The Pauli gate before the Clifford gate (A) can be delayed after Clifford gate (B). In (B), the delayed Pauli gate is combined with the Pauli gates inside the Clifford gate and changed to a new Pauli operation.

TABLE 3 Rules for performing Pauli frames for $CONT_L$

C_{in}	T_{in}	C_{out}	T_{out}
X_L	I_L	$X_L Z_L^{a+c} \lvert\psi_L\rangle$	$X_L X_L^b \lvert\psi_L\rangle = X_L^{b+1} \lvert\psi_L\rangle$
I_L	X_L	$I_L Z_L^{a+c} \lvert\psi_L\rangle$	$X_L X_L^b \lvert\psi_L\rangle = X_L^{b+1} \lvert\psi_L\rangle$
Z_L	I_L	$Z_L Z_L^{a+c} \lvert\psi_L\rangle = Z_L^{a+c+1} \lvert\psi_L\rangle$	$I_L X_L^b \lvert\psi_L\rangle$
I_L	Z_L	$Z_L Z_L^{a+c} \lvert\psi_L\rangle = Z_L^{a+c+1} \lvert\psi_L\rangle$	$Z_L X_L^b \lvert\psi_L\rangle$

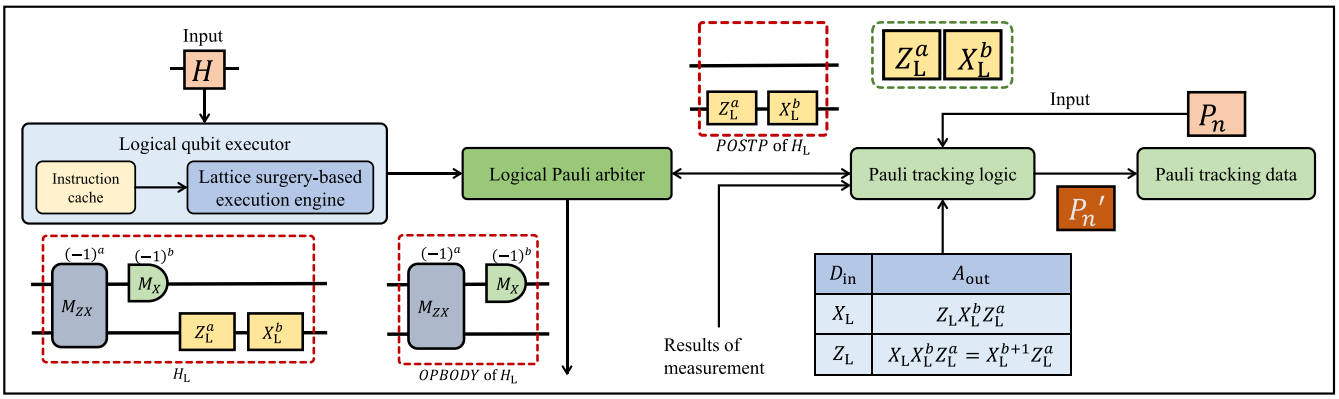

FIGURE 6 Schematic diagram of the operation processing flow of the Logical Qubit Layer

TABLE 4 Logical qubit layer operation rules

Operations	Execution steps
Initialization to $\lvert 0_L\rangle$	1. Initialize target logical qubit to $\lvert 0_L\rangle$. 2. Initialize target physical qubits to $\lvert 0\rangle$. 3. Set logical Pauli frame of target logical qubit to I_L. 4. Set physical Pauli frame of target physical qubits to I.
Measurement	1. Measure target physical qubit. 2. Correct measurement result based on physical Pauli frame. 3. Translate physical Pauli frame to logical Pauli frame 4. Correct result based on logical Pauli frame
Pauli gates	1. Map logical Pauli frame of target logical qubit.
Clifford gates	1. Map logical Pauli frame of target logical qubit. 2. Apply Clifford gate on target logical qubit.
Non-Clifford gates	1. Flush Pauli frame of target logical qubit. 2. Apply non-Clifford gate on target logical qubit.

In our Pauli frame tracking architecture separated into two layers, the physical-level Pauli frame structure performs only error correction of physical qubits according to the decoding process after the ESM.

Figure 7 shows the schematic execution procedure and functions of the *Physical Qubit Layer*, with the following detailed flow description:

- The *QEC Cycle Generator* generates ESM commands scheduled by the *Logical Qubit Executor*. The results from the execution of the ESM command are used as input to the *QEC Decoder*, which performs the error decoding process.
- The *QEC Decoder* detects the error using error syndromes measured as many times as the code distance. It also generates physical-level Pauli operations that signal an error correction if necessary.
- The *Physical Qubit Executor* decomposes the *OPBODY* of the logical operation into physical operations. It also schedules the physical operations for all physical qubits constituting the logical qubit.
- The *Physical Pauli Arbiter* executes physical operations for each physical qubit according to the rules in Table 5.
- The *Physical Pauli Frame Unit* stores Pauli operations according to the existing Pauli frame technique. When the measurement of a physical qubit is requested, this unit compensates the obtained measurement result with Pauli tracking data and returns it.

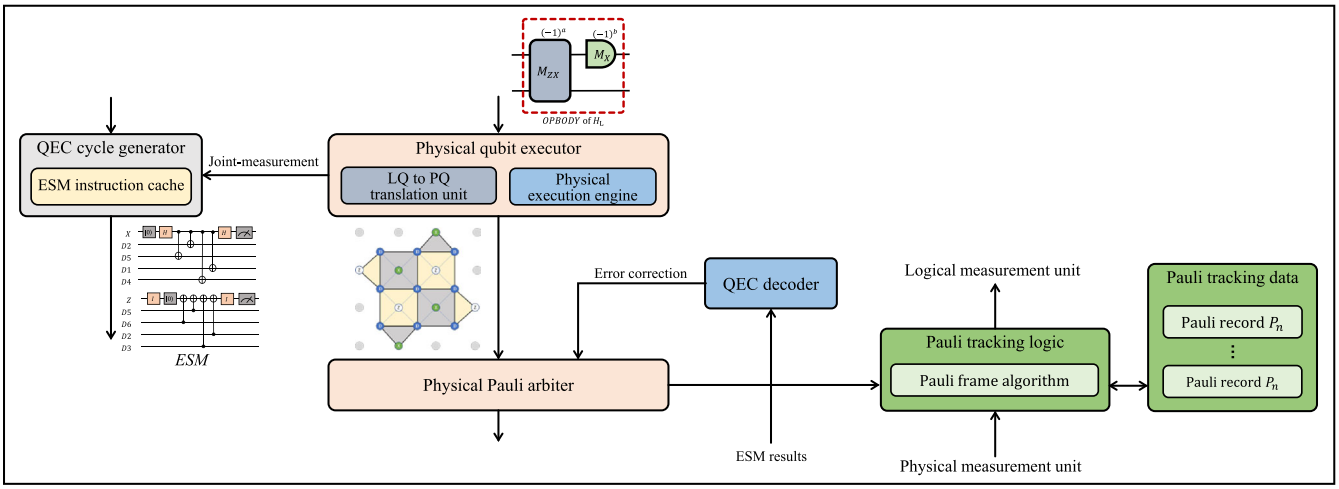

FIGURE 7 Schematic diagram of the operation processing flow of the Physical Qubit Layer

TABLE 5 Physical qubit layer operation rules

Operations	Execution steps
Initialization to $\|0\rangle$	1. Initialize target physical qubits to $\|0\rangle$. 2. Set physical Pauli frame of target physical qubits to *I*.
Measurement	1. Measure target physical qubit. 2. Correct measurement result based on physical Pauli frame
Pauli gates	1. Map physical Pauli frame of target physical qubit.
Clifford gates	1. Map physical Pauli frame of target physical qubit. 2. Apply Clifford gate on target physical qubit.
Non-Clifford gates	1. Flush Pauli frame of target physical qubit. 2. Apply non-Clifford gate on target physical qubit.

4.4 | Measurement operations

Compensating measurement operation results with physical- and logical-level Pauli frames makes the logical measurement operation results correct. As shown in Figure 8, a correct logical measurement result is obtained through the three following steps.

Step1. *Physical qubit measurement*: The measurement result of a physical qubit is corrected using the physical-level Pauli frame of the corresponding qubit. The fixed value becomes the final result of the physical measurement in the *Physical Qubit Layer*.

Step2. *Logical measurement conversion*: Because a logical operator is composed of multiple physical operators, combining physical qubit measurement data and converting them into a single logical qubit measurement value is necessary.

Step3. *Logical value compensation*: The final corrected logical measurement value is obtained by combing the previous converted logical qubit measurement value with Pauli frame data.

For example, Surface Code-17 can determine the measurement value of a logical qubit through the steps above after measuring the nine physical data qubits constituting a logical qubit.

5 | PERFORMANCE ANALYSIS

In this section, we analyze quantitative performance and experimental results to verify the efficiency of our multi-layered Pauli tracking architecture.

The efficiency of the proposed architecture depends on the internal structure of logical operations making up logical operators. For example, a typical lattice surgery technique consists of Joint-Measurement, logical qubit measurement, and Pauli operations for logical operation correction. Because the logical-level Pauli frame unit can eliminate internal Pauli operations in a logical operator, the efficiency of the proposed architecture is determined by the ratio of physical operations constituting logical Pauli operations to the total physical operations. Because the results of the Joint-Measurement/Measurement operations determine the actual execution of a Pauli operator, the efficiency of the proposed architecture on a logical operator can also be determined according to the reduction rate, as presented in Table 6.

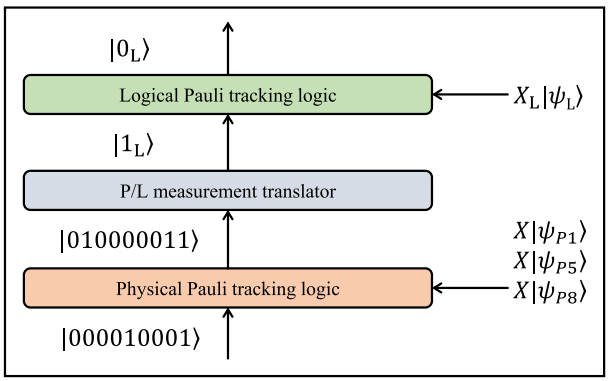

FIGURE 8 The process of obtaining the correction measurement result of logical qubit, starting from the measurement results of many physical qubits

The performance of the logical-level Pauli frame architecture significantly depends on the ratio and order of the operators composing the circuit and the random measured value of the Joint-Measurement operation. Therefore, we compared the proposed Pauli tracking architecture's performance considering the characteristics of the target systems. Overall, we divided the comparison system into three categories: a system applying only physical-level Pauli frames to physical Pauli operators, a system applying logical-level Pauli frames to logical Pauli operators, and our system, which applies logical-level Pauli frames to both logical Pauli and Clifford operators. All three system categories were benchmarked for the same quantum circuit, and the number of actual physical operators executed for each system was compared.

The proposed architecture's performance is determined on the basis of the qubit error rate since a physical-level Pauli frame is applied, as in previous studies [16, 17, 26, 29, 30]. Therefore, we do not include an efficiency analysis in the Physical Qubit Layer but only compare the number of physical operations generated for the logical circuit of the Logical Qubit Layer.

As described above, the architecture performance of the Pauli frame is significantly affected by the composition ratio of operations constituting the quantum circuit. Therefore, analyzing the composition ratio of quantum operators in many quantum circuits is required to evaluate the accurate architecture performance in comparison targets. After studying the benchmark circuit of ScaffCC [31] in Riesebos and others [25], we obtained the result confirming the composition ratio of gates constituting each circuit. The analysis results in [25] revealed that approximately 6% of the entire circuit is occupied by the Pauli gate, 20%–50% by the non-Clifford gate, and the rest by the Clifford gate. Based on the above ratio, the gate composition ratio of the benchmarking quantum circuit was calculated for efficiency analysis of the proposed Pauli tracking architecture, as follows:

- Pauli gate: 6%
- Clifford gate: 54%
- Non-Clifford gate: 40%

We implemented a performance analysis program to generate random circuits according to a defined ratio through the IBM Qiskit library [32]. Then, the average values of the execution results of each system were compared. The test procedure for benchmarking was the same as TEST PROCEDURE 1.

TEST PROCEDURE 1

Input D_{max}: the number of max circuit depth
C_L: set of Categories $\{C_1, C_2, C_3\}$
Q_L: set of logical qubits $\{Q_0, \ldots, Q_n\}$

Output R_n: results of execution by the categories C_L

Circuit depth $D_i \leftarrow 1$
while $D_i \le D_{max}$ **do**
  $D_T \leftarrow$ generate arbitrary circuit with depth D_i
  while $D_T \neq \varnothing$ **do**
    $G_i \leftarrow$ pop a gate that is first in circuit D_T
    R_1 += calculate the number of physical operations after applying the Category C_1 to the logical operation G_i of qubit Q_i
    R_2 += calculate the number of physical operations after applying the Category C_2 to the logical operation G_i of qubit Q_i
    R_3 += calculate the number of physical operations after applying the Category C_3 to the logical operation G_i of qubit Q_i
  $D_i = D_i + 1$

C_1: System category with physical-level Pauli frames
C_2: System category with logical-level Pauli frames for logical Pauli operators
C_3: System category with logical-level Pauli frames for both logical Pauli operators and Clifford operators (Our architecture)

Figure 9 shows the performance comparison results obtained from the system benchmarking. As shown in the figure, if the circuit has few Pauli operators and significant non-Clifford operators, the probability of encountering Pauli operators before executing non-Clifford operators is very low. The efficiency improvement caused by the Pauli frame architecture was less than 1%. However, we confirmed that physical gates generated in category 3 were approximately 5% fewer than those generated in category 1 or 2.

In general, reducing the number of physical operations using logical-level Pauli frames is critical for the efficient implementation of fault-tolerant quantum computing. As described in the performance comparison experiment, the difference in physical operators between

TABLE 6 Reduced gates with Logical Pauli Frame Unit

Operations	Reduced operations	Reduction rate
I_L, X_L, Z_L, Y_L	All Pauli operations are mapped to logical Pauli frames.	$\frac{N_g(P_i)}{N_g(Total)} \times 100$
H_L	The ratio is determined by the Joint-Measurement results (a, b)	$\frac{N_g(P_i^a)+N_g(P_i^b)}{N_g(Total)} \times 100$
$CNOT_L$	The ratio is determined by the Joint-Measurement results (a, b, c)	$\frac{N_g(P_i^{a+c})+N_g(P_i^b)}{N_g(\mathrm{Total})} \times 100$
Magic State Distillation	The Pauli operation generated inside Multitarget $CNOT_L$ and H_L is stored in the Pauli frame, but the total number of operations does not change as all operations are applied before measurement. However, as the actual operation is not executed in the physical layer, resource efficiency is improved.	0
S_L	The ratio is determined by the Joint-Measurement results (a, b, c, d)	$\frac{N_g(P_i^{a+b+c+d})}{N_g(\mathrm{Total})} \times 100$
T_L	The ratio of the logical Pauli-Z operation is determined by the Joint-Measurement result (a_T, c_T), and the execution of the S operation is determined by another measurement value (b_T, d_T). Therefore, the efficiency of Pauli frames is determined according to the values of the measured values.	If $d_T + b_T$ is even, $\frac{N_g(P_i^{a_T+c_T})}{N_g(\mathrm{Total})} \times 100$ If $d_T + b_T$ is odd, $\frac{N_g(P_i^{a+b+c+d+a_T+c_T})}{N_g(\mathrm{Total})} \times 100$

Note: N_g: Number of physical operations that perform internal Pauli operations. $P_i \in \{I_L, X_L, Z_L, Y_L\}$: Internal Pauli operations. $a, b, c, d, a_T, b_T, c_T, d_T \in \{0,1\}$: Joint-Measurement results. *Total*: Number of physical operations that perform logical operation.

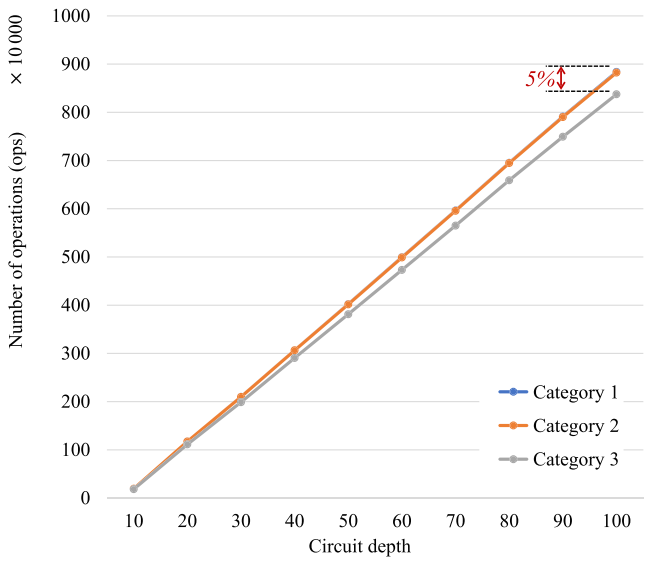

FIGURE 9 Performance analysis through comparison of benchmarking results of qubit circuits

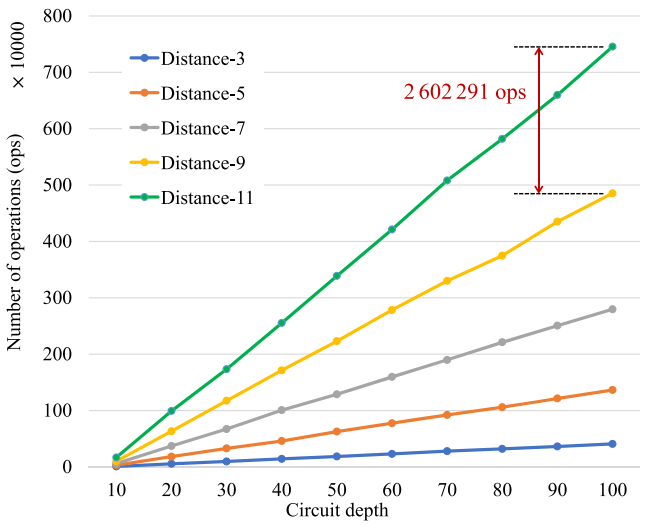

FIGURE 10 Difference in the number of generated physical operations according to the surface code distance

our proposed Pauli frame architecture and the existing one was within 5%. However, as the size of the surface code, where a logical operator comprises a combination of many physical operations, equals the code distance, the effect of reducing the number of logical operators in the real system is generally tremendous. The surface code can reduce the error rate by increasing its code distance. However, the number of physical gates to be executed increases exponentially with increasing code distance. If we increase the code distance of the identical circuit from 3 to 11, the effectiveness of the logical-level Pauli frames significantly increases. Figure 10 shows the difference in the number of physical operations generated by the proposed logical-level Pauli frame architecture as the code distance of the benchmark circuit increases. As the code distance of the surface code gradually increases, the reduction effect of physical operators using logical-level Pauli frames becomes more significant.

6 | CONCLUSIONS

The quantum software layer encodes multiple physical qubits into one logical qubit and provides fault tolerance for errors in the physical qubits through this encoding process. As the code distance constituting logical qubits increases, the operation of physical qubits constituting logical qubits increases exponentially. To provide efficient logical qubits, a method for minimizing physical operations in actual quantum hardware must be presented.

Analyzing the operation characteristics of logical operators in the lattice surgery, we proposed a multilayered Pauli tracking architecture that effectively processes logical operators using the Pauli frame technique in logical qubits based on the lattice surgery method. Our architecture consists of rules for processing internal Pauli operations included in logical operators as a Pauli frame and a processing structure divided into two layers. The advantage of this study is that it can reduce the amount of information that needs to be processed in software and reduce physical qubit operations through a processing structure separated into logical–physical layers. The two simulations in Section 5 showed that the number of physical operations was reduced by approximately 5% compared with the existing Pauli frame methods, and the importance of the logical Pauli frames was confirmed through an experiment to increase the code distance for the same circuit. Because the code distances constituting logical qubits will increase in fault-tolerant quantum computing, our comparison results indicate that our proposal can be effectively applied in future quantum computing.

However, our study has the following limitations. Our method is limited to lattice surgery operations on rotated surface codes. In addition, it does not include all operation methods of lattice surgery and is described based on a typical quantum circuit of a general-purpose operator. Therefore, implementations and detailed methods of other methods may differ, and the performance analysis results may vary accordingly. Our architecture and data flow are focused only on how Pauli frames are processed and therefore do not include all components and data flows of the quantum control layer. To provide large-scale, fault-tolerant quantum computing, it is necessary to integrate data flow and interworking methods with multiple components that are not included in the architecture, and a study on effective Pauli frame processing method for the latest lattice surgery-based surface code should be performed.

ACKNOWLEDGMENT

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).

CONFLICT OF INTEREST

The authors declare that there are no conflicts of interest.

ORCID

Jin-Ho On https://orcid.org/0000-0001-7610-2204

REFERENCES

1. J. Preskill, *Quantum computing in the Nisq era and beyond*, Quantum. **2** (2018), 79.
2. I. Kang, J. Y. Choung, D.i. Kang, and I. Park, *Divergence of knowledge production strategies for emerging technologies between late industrialized countries: focusing on quantum technology*, ETRI Journal. **43** (2021), no. 2, 246–259.
3. J. Preskill, *Reliable quantum computers*, Proc. Royal Soc. London. Ser. A: Math. Phys. Eng. Sci. **454** (1998), no. 1969, 385–410.
4. S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven, *Characterizing quantum supremacy in near-term devices*, Nat. Phys. **14** (2018), no. 6, 595–600.
5. B. M. Terhal, *Quantum error correction for quantum memories*, Rev. Mod. Phys. **87** (2015), no. 2, 307–346.
6. E. T. Campbell, B. M. Terhal, and C. Vuillot, *Roads towards fault-tolerant universal quantum computation*, Nature **549** (2017), no. 7671, 172–179.
7. F. Gaitan, *Quantum error correction and fault tolerant quantum computing*, CRC Press Boca Raton, FL, 2008.
8. D. A. Lidar and T. A. Brun, *Quantum error correction*, Cambridge university press, 2013.
9. D. Gottesman, *An introduction to quantum error correction and fault-tolerant quantum computation, in quantum information science and its contributions to mathematics*, (Proc. Symp. Appl. Math.), 2010, pp. 13–58.
10. A. Y. Kitaev, *Fault-tolerant quantum computation by Anyons*, Ann. Phys. Rehabil. Med. **303** (2003), no. 1, 2–30.
11. 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), no. 3, 032324.
12. H. Bombín, *Topological order with a twist: Ising Anyons from an abelian model*, Phys. Rev. Lett. **105** (2010), no. 3, 030403.
13. J. Roffe, *Quantum error correction: an introductory guide*, Contemporary Physics. **60** (2019), no. 3, 226–245.
14. D. Litinski, *A game of surface codes: Large-scale quantum computing with lattice surgery*, Quantum. **3** (2019), 128.
15. E. Knill, *Quantum computing with realistically noisy devices*, Nature **434** (2005), no. 7029, 39–44.
16. P. Aliferis and J. Preskill, *Fault-tolerant quantum computation against biased noise*, Phys. Rev. A **78** (2008), no. 5, 052331.
17. D. P. DiVincenzo and P. Aliferis, *Effective fault-tolerant quantum computation with slow measurements*, Phys. Rev. Lett. **98** (2007), no. 2, 020501.

18. N. C. Jones, R. Van Meter, A. G. Fowler, P. L. McMahon, J. Kim, T. D. Ladd, and Y. Yamamoto, *Layered architecture for quantum computing*, Physical Review X. **2** (2012), no. 3, 031007.
19. C. Horsman, A. G. Fowler, S. Devitt, and R. V. Meter, *Surface code quantum computing by lattice surgery*, New J. Phys. **14** (2012), no. 12, 123011.
20. H. Bombín and M. A. Martin-Delgado, *Optimal resources for topological two-dimensional stabilizer codes: comparative study*, Phys. Rev. a **76** (2007), no. 1, 012305.
21. D.S. Wang, A.G. Fowler, A.M. Stephens, and L.C.L. Hollenberg, *Threshold error rates for the toric and surface codes*, arXiv preprint arXiv:0905.0531, 2009.
22. H. Bombin, R. S. Andrist, M. Ohzeki, H. G. Katzgraber, and M. A. Martin-Delgado, *Strong resilience of topological codes to depolarization*, Phys. Rev. X. **2** (2012), no. 2, 021004.
23. A. M. Stephens, *Fault-tolerant thresholds for quantum error correction with the surface code*, Phys. Rev. a **89** (2014), no. 2, 022321.
24. K. Fujii, *Quantum computation with topological codes: from qubit to topological fault-tolerance*, Springer, 2015.
25. (L. Riesebos, X. Fu, S. Varsamopoulos, C.G. Almudever, and K. Bertels, *Pauli frames for quantum computer architectures*, (Proceedings of the 54th Annual Design Automation Conference, Austin, TX, USA), 2017. pp. 1–6.
26. D. Gottesman, *The Heisenberg representation of quantum computers*, arXiv preprint, 1998. https://doi.org/10.48550/arXiv.quant-ph/9807006
27. X. Fu, L. Lao, K. Bertels, and C. G. Almudever, *A control microarchitecture for fault-tolerant quantum computing*, Microprocess. Microsyst. **70** (2019), 21–30.
28. M. A. Nielsen and I. Chuang, *Quantum computation and quantum information*, Cambridge university press, 2000.
29. S. Aaronson and D. Gottesman, *Improved simulation of stabilizer circuits*, Phys. Rev. A **70** (2004), no. 5, 052328.
30. C. Chamberland, P. Iyer, and D. Poulin, *Fault-tolerant quantum computing in the Pauli or Clifford frame with slow error diagnostics*, Quantum. **2** (2018), 43.
31. A. JavadiAbhari, S. Patil, D. Kudrow, J. Heckey, A. Lvov, F.T. Chong, and M. Martonosi, *Scaffcc: a framework for compilation and analysis of quantum computing programs*, (Proceedings of the 11th ACM Conference on Computing Frontiers, Cagliari, Italy), 2014. pp. 1–10.
32. A. Cross, *The IBM Q experience and Qiskit open-source quantum computing software, in APS march*, Meeting Abstracts. 2018, L58. 003.
33. S. Bravyi and A. Kitaev, *Universal quantum computation with ideal Clifford gates and Noisy Ancillas*, Phys. Rev. A **71** (2005), no. 2, 022316.

AUTHOR BIOGRAPHIES

Jin-Ho On received his MS and PhD degrees in Computer Engineering in 2008 and 2012, respectively, at the Jeonbuk National University, Jeonju, Republic of Korea. He has been working at the Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea since 2013. There, he has contributed to the development of the energy-aware operating system, microservices architecture for AI computing, and quantum operating system for fault-tolerant quantum computing. Currently, he works as a senior researcher. His current research interests include Quantum computing architecture and fault-tolerant quantum operation systems.

Chei-Yol Kim received his MS and PhD degrees in Electronics Engineering in 2001 and 2017, respectively, from the Kyungpook National University, Daegu, Republic of Korea. He joined Electronics and Telecommunications Research Institute in 2001. Currently, he is a principal researcher at their Future Computing Research Division. He has worked in the field of server virtualization, Linux kernel development, cluster filesystem, and microfunction services. Currently, his research focuses on Quantum computing areas, including quantum computing OS, system SW, and fault-tolerance quantum computing architecture.

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 the LG Multimedia Research Laboratory. Since 2005, he has been working as a principal researcher at the Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea. His current research interests are in quantum computing and cloud systems.

Sang-Min Lee earned her BS degree in Computer Engineering at the 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 developing the SCSI and FC RAID system, distributed parallel file system, dual-mode big data platform, and simulation technology for the digital twin. Currently, she is working as a principal researcher. Her current research interests include distributed systems, extreme storage systems, and quantum operating systems.

Gyu-Il Cha received his BS and MS degrees in Computer Science in 1998 and 2000, respectively, at the 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 interest includes developing a quantum operating system for fault-tolerant quantum computing. He has been involved in the technology development of operating systems, memory virtualization, supercomputing, microservice architectures, and extreme storage systems.

How to cite this article: J.-H. On, C.-Y. Kim, S.-C. Oh, S.-M. Lee, and G.-I. Cha, *A multilayered Pauli tracking architecture for lattice surgery-based logical qubits*, ETRI Journal **45** (2023), 462–478. https://doi.org/10.4218/etrij.2022-0037

APPENDIX A: RULES FOR APPLYING LOGICAL-LEVEL PAULI FRAMES

We will further describe general-purpose gates for processing quantum circuits using logical qubits.

A.1 | Logical move

Logical *MOVE* operation, the simplest form of the lattice surgery technique, includes Joint-Measurement implemented as *Merge/Split*, as shown in Figure A1, measurement of logical qubits, and *Z*- or *X*-POST operations for correcting the logical operator using measured values. According to the *OPBODY* measurement value, the *POSTP* operation can be processed using rules in Table A1.

A.2 | Logical H gate

As shown in Figure A2, in logical *H* operation, M_{XX} operation of $MOVE_{\mathrm{L}}$ operation is changed to M_{ZX}, and the same operations as the $MOVE_{\mathrm{L}}$ operation are executed in the *POSTP* part. Table A2 shows the execution rules of the logical-level Pauli frames.

A.3 | Logical Multitarget CNOT

Logical *Multitarget CNOT* in the lattice surgery technique is implemented as the circuit in Figure A3. The single target of the two-qubit $CNOT_{\mathrm{L}}$ is extended to multiple targets, and *Merge/Split* is executed on the targets, forming an entanglement. It is a structure controlled through the M_{ZZ} of the control. Table A3 shows the execution rules of the logical-level Pauli frames for the *multitarget* $CNOT_{\mathrm{L}}$.

A.4 | Magic State Distillation

There are various lattice surgery-based methods for *Magic State Distillation* [33]. Figure A4 represents a representative 15:1 Magic State Distillation circuit. For 15 logical qubits, $|Y_L\rangle$ and $|A_L\rangle$ states required by S_{L} and T_{L} gates are inputs, respectively, and the distillation process is performed. The internal circuit is configured in an entangled state through five logical *CNOT*s and four logical *H* gates.

The circuit is repeatedly executed until the desired result is obtained according to the measurement result. As shown in Figure A4, logical-level Pauli frames can delay *POSTP* in all logical operations until measurement; the distillation step, which accounts for a very high proportion among the components of quantum circuits, can be efficiently processed through *POSTP* delay.

A.5 | Logical S gate

The logical *S* gate, a Clifford gate, is implemented through the $CNOT_{\mathrm{L}}$ operation, and the *POSTP* for the $|Y_{\mathrm{L}}\rangle$ state is created through *state injection* and *Magic State Distillation*, as shown in Figure A5. The $CNOT_{\mathrm{L}}$ in the S_{L} gate can be reconstructed by decomposing Pauli operations, as shown in Figure A5B. Table A5 shows the execution rules for logical-level Pauli frames for the S_{L} gate.

A.6 | Logical T gate

Figure A6 depicts the structure of the logical *T* gate: a non-Clifford operation. Owing to the relationship between Pauli frames and non-Clifford operations, logical-level Pauli frames must physically perform all operations of a Pauli frame before the T_{L} gate operation. A T_{L} gate operation based on lattice surgery includes *POSTP*, where the execution of the S_{L} gate is internally determined according to the measured value of the

TABLE A1 Rules for performing Pauli frames for $MOVE_{\mathrm{L}}$

D_{in}	A_{out}
X_{L}	$X_{\mathrm{L}}X_{\mathrm{L}}{}^{b}Z_{\mathrm{L}}{}^{a}\lvert\psi_{\mathrm{L}}\rangle = X_{\mathrm{L}}{}^{b+1}Z_{\mathrm{L}}{}^{a}\lvert\psi_{\mathrm{L}}\rangle$
Z_{L}	$Z_{\mathrm{L}}X_{\mathrm{L}}{}^{b}Z_{\mathrm{L}}{}^{a}\lvert\psi_{\mathrm{L}}\rangle$

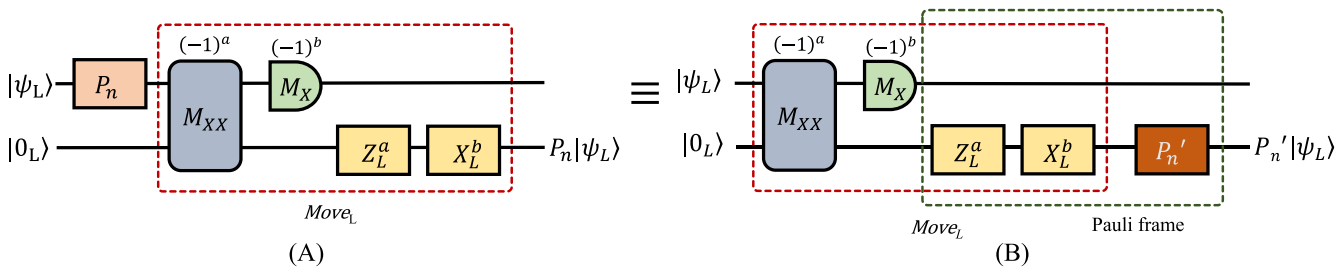

FIGURE A1 How to handle Pauli frames for lattice surgery-based *MOVE* operation

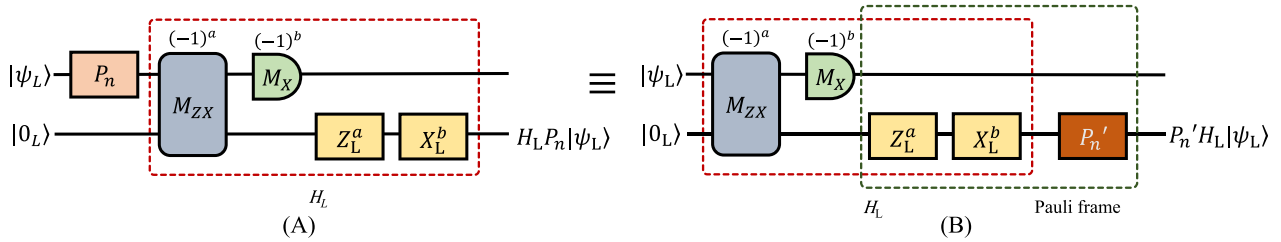

FIGURE A2 How to handle Pauli frames for lattice surgery-based H operation

TABLE A2 Rules for performing Pauli frames for H_L

D_{in}	A_{out}
X_L	$Z_L X_L{}^b Z_L{}^a \|\psi_L\rangle$
Z_L	$X_L X_L{}^b Z_L{}^a \|\psi_L\rangle = X_L{}^{b+1} Z_L{}^a \|\psi_L\rangle$

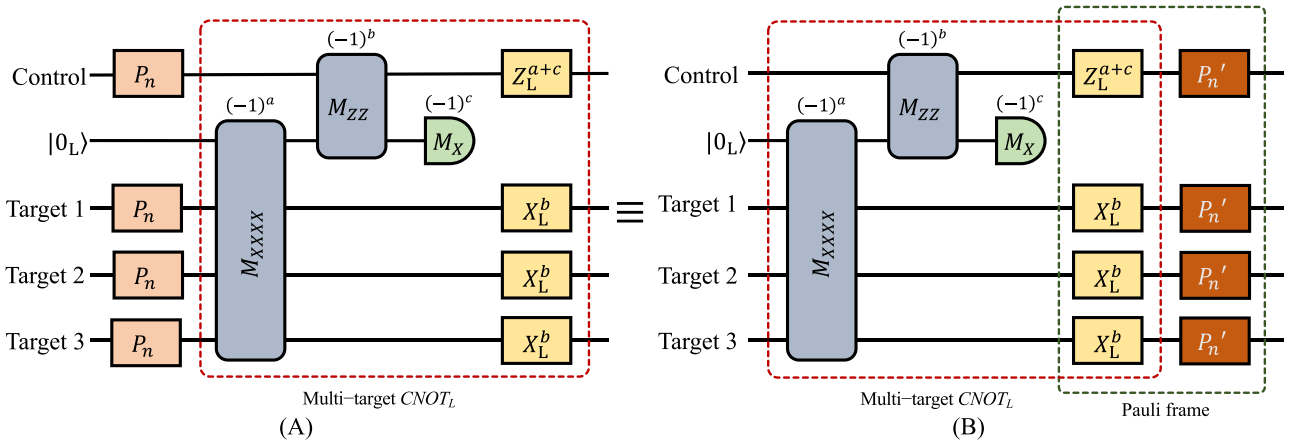

FIGURE A3 How to handle Pauli frames for lattice surgery-based multi-target CNOT operation

TABLE A3 Rules for performing Pauli frames for *multitarget* $CNOT_L$

C_{in}	$T1_{in}$	$T2_{in}$	$T3_{in}$	C_{out}	$T1_{out}$	$T2_{out}$	$T3_{out}$
X_L	I_L	I_L	I_L	$X_L Z_L{}^{a+c} \|\psi_L\rangle$	$X_L X_L{}^b \|\psi_L\rangle = X_L{}^{b+1} \|\psi_L\rangle$	$X_L X_L{}^b \|\psi_L\rangle = X_L{}^{b+1} \|\psi_L\rangle$	$X_L X_L{}^b \|\psi_L\rangle = X_L{}^{b+1} \|\psi_L\rangle$
I_L	X_L	X_L	X_L	$I_L Z_L{}^{a+c} \|\psi_L\rangle$	$X_L X_L{}^b \|\psi_L\rangle = X_L{}^{b+1} \|\psi_L\rangle$	$X_L X_L{}^b \|\psi_L\rangle = X_L{}^{b+1} \|\psi_L\rangle$	$X_L X_L{}^b \|\psi_L\rangle = X_L{}^{b+1} \|\psi_L\rangle$
Z_L	I_L	I_L	I_L	$Z_L Z_L{}^{a+c} \|\psi_L\rangle = Z_L{}^{a+c+1} \|\psi_L\rangle$	$I_L X_L{}^b \|\psi_L\rangle$	$I_L X_L{}^b \|\psi_L\rangle$	$I_L X_L{}^b \|\psi_L\rangle$
I_L	Z_L	Z_L	Z_L	$Z_L Z_L{}^{a+c} \|\psi_L\rangle = Z^{a+c+1} \|\psi_L\rangle$	$Z_L X_L{}^b \|\psi_L\rangle$	$Z_L X_L{}^b \|\psi_L\rangle$	$Z_L X_L{}^b \|\psi_L\rangle$

M_{XX}/M_{ZZ} operation, as shown in Figure A6. If the execution of the S_L gate is not selected according to the value of $d_T + b_T$, $Z_L{}^{a_T+c_T}$ is executed. However, if the execution of the S_L gate is selected, as shown in Figure A6B-1, *POSTP* $Z_L{}^{a_T+c_T}$ before the S_L gate is not executed immediately. *POSTP* $Z_L{}^{a_T+c_T}$ before the S_L gate can comprise a Pauli frame through Clifford gate operations and can therefore be combined with the Pauli operations in the S_L gate, as shown in Figure A6B-2.

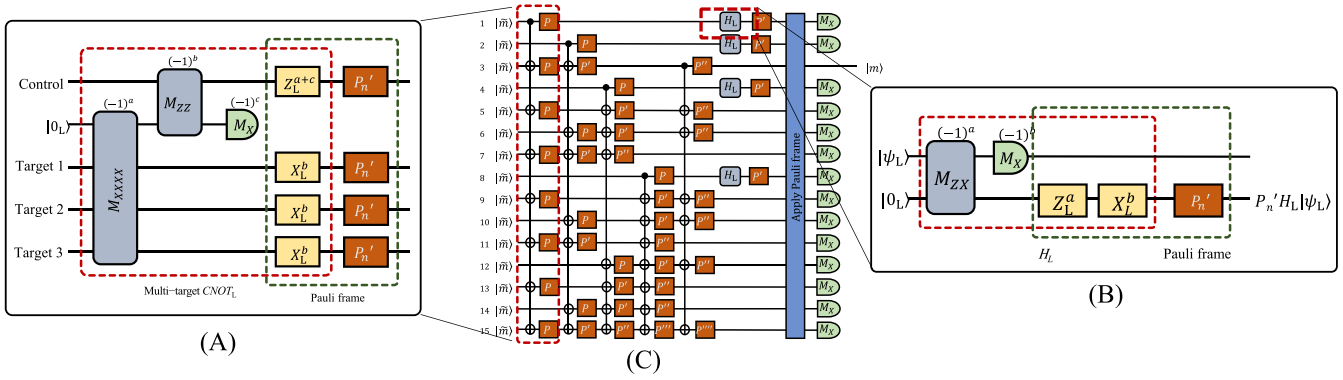

FIGURE A4 How to handle Pauli flames for lattice surgery-based Magic State Distillation

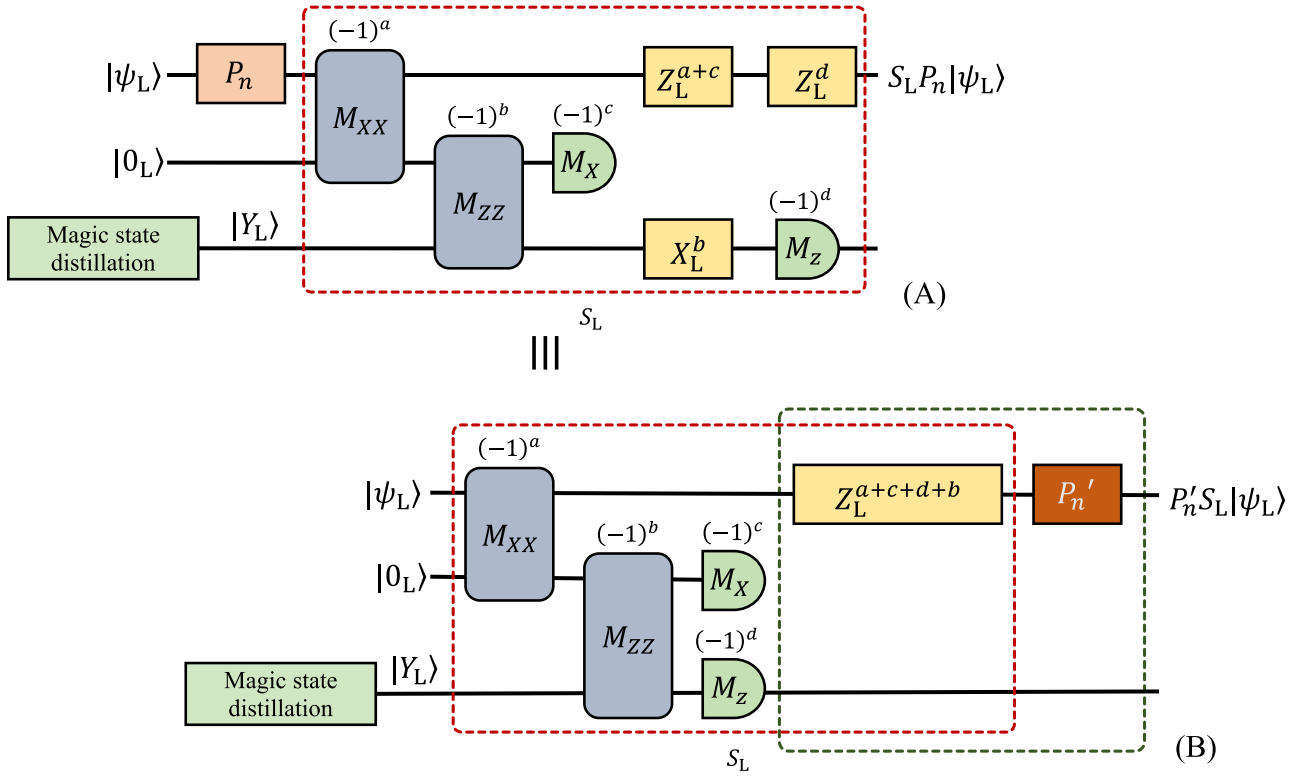

FIGURE A5 How to handle Pauli frames for lattice surgery-based S operation

TABLE A5 Rules for performing Pauli frames for S_L

D_{in}	D_{out}
X_L	$Y_L Z_L^{d+b} Z_L^{a+c} \lvert\psi_L\rangle = Y_L Z_L^{a+c+d+b} \lvert\psi_L\rangle$
Z_L	$Z_L Z_L^{d+b} Z_L^{a+c} \lvert\psi_L\rangle = Z_L^{(a+c+d+b)+1} \lvert\psi_L\rangle$

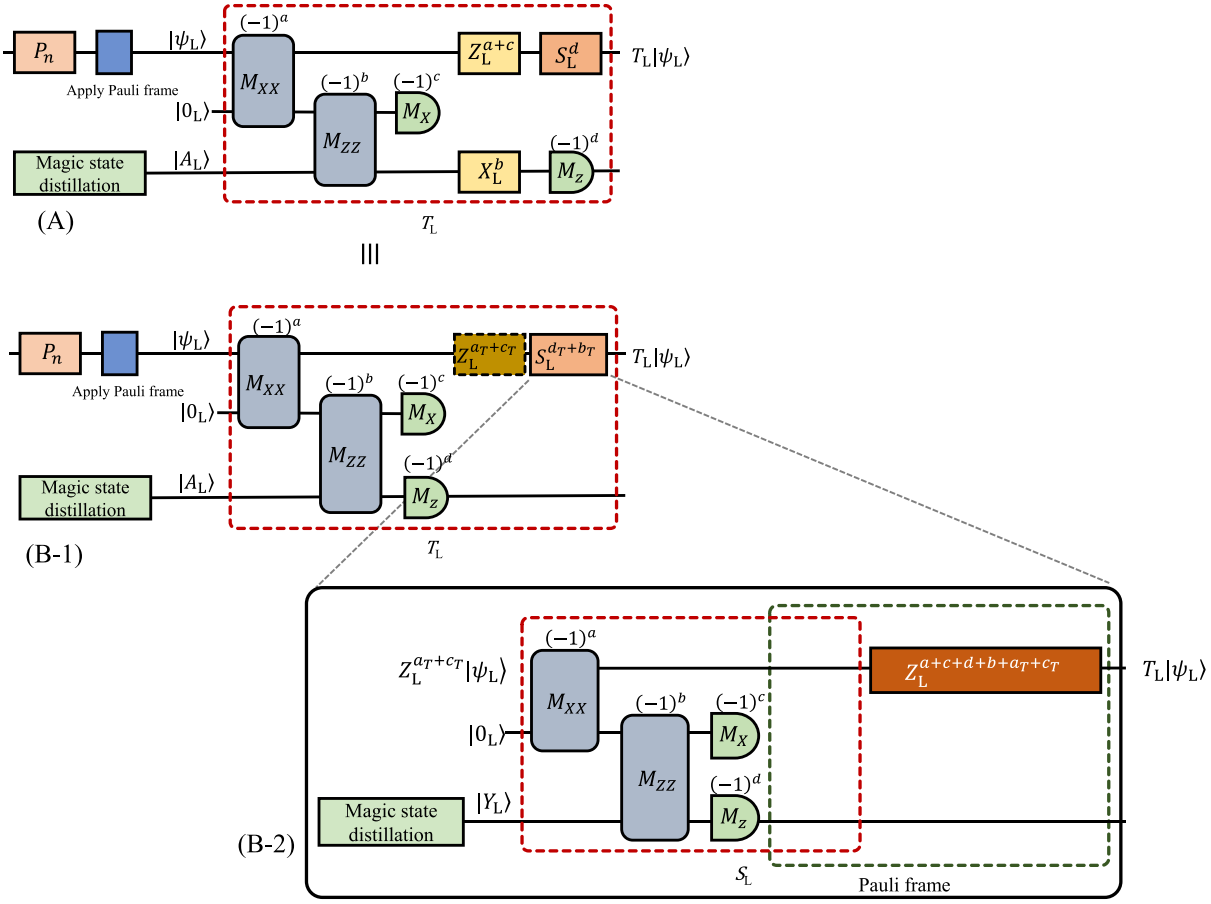

FIGURE A6 How to handle Pauli frames for lattice surgery-based T operation