Dense packing of the surface code: Code deformation procedures and hook-error-avoiding gate scheduling

Kohei Fujiu,[1] Shota Nagayama,[2] Shin Nishio,[1,3] Hideaki Kawaguchi,[1] and Takahiko Satoh[4]
[1]*Graduate School of Science and Technology, Keio University, Yokohama, Kanagawa 223-8522, Japan*
[2]*Graduate School of Media Design, Keio University, Yokohama, Kanagawa 223-0061, Japan*
[3]*Department of Physics and Astronomy, University College London, London WC1E 6BT, United Kingdom*
[4]*Faculty of Science and Technology, Keio University, Yokohama, Kanagawa 223-8522, Japan*

(Received 9 November 2025; revised 29 January 2026; accepted 24 February 2026; published 3 April 2026)

The surface code is one of the leading quantum error correction codes for realizing large-scale fault-tolerant quantum computing (FTQC). One major challenge in realizing surface-code-based FTQC is the extremely large number of qubits required. To mitigate this problem, fusing multiple code words of the surface code into a densely packed configuration has been proposed. It is known that by using dense packing, the number of physical qubits required per logical qubit can be reduced to approximately three-fourths compared to simply placing surface-code patches side by side. Despite its potential, concrete deformation procedures and quantitative error-rate analyses have remained largely unexplored. In this work, we present a detailed code-deformation procedure that transforms multiple standard surface-code patches into a densely packed, connected configuration, along with a conceptual microarchitecture to utilize this dense packing. We also propose a cnot gate scheduling for stabilizer measurement circuits that suppresses hook errors in the densely packed surface code. We performed circuit-level Monte Carlo noise simulation of densely packed surface codes using this gate scheduling. The numerical results demonstrate that as the code distance of the densely packed surface code increases and the physical error rate decreases, the logical error rate of the densely packed surface code becomes lower than that of the standard surface code. Furthermore, we find that only when employing hook-error-avoiding syndrome extraction can the densely packed surface code achieve a lower logical error rate than the standard surface code, while simultaneously reducing the space overhead.

DOI: 10.1103/7lm4-3bnh

I. INTRODUCTION

A Quantum computer is expected to solve many important classically intractable problems [1–3]. However, the performance of a quantum computer is fundamentally constrained by the coherence times of its physical qubits [4]. To reliably store and manipulate quantum information in the presence of noise, quantum error-correcting codes, which encode logical quantum information onto redundant physical qubits, have been proposed [5–7].

Topological quantum error-correcting codes [7,8] have attracted particular attention, as they require only the measurement of spatially local operators. This property makes them compatible with the two-dimensional solid-state qubit architectures such as superconducting qubits [9], and spin qubits [10]. Within this class, the surface code [11,12] stands out as one of the most promising codes. For fault-tolerant quantum computation with the surface code in two-dimensional architectures, logical gate implementation techniques such as lattice surgery [13] and defect braiding [12] have been proposed. Lattice surgery has attracted particular attention because it enables computation with much lower resource overhead [14–16]. In lattice surgery-based computation, multiple patches of surface code are arranged and manipulated [13,14]. Here, a patch refers to an independent code word of surface code with boundaries, encoding a single logical qubit. In this work, we specifically treat a patch as a rotated surface code [17–20].

One major drawback of the surface code is its significant demand for a large number of physical qubits per logical qubit. The number of physical qubits required to construct a single patch of the surface code increases proportionally to the square of the code distance. To alleviate this issue, several schemes have been investigated to reduce the physical-qubit overhead per a code word of the surface code [2,21–23].

In this work, we focus on dense packing [2], which fuses multiple patches of the surface code and stores their code words at higher density. This method has been proposed to reduce the number of physical qubits required per logical qubit to approximately three-quarters. This method could contribute to alleviating the physical qubit requirements in surface-code-based fault-tolerant quantum computation. The dense packing is expected to potentially introduce a memory hierarchy in fault-tolerant quantum computing [24–27] and reduce its spatial overhead. However, methods for transitioning from the state of a standard surface code to a densely packed

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

state, as well as the logical error rates in a densely packed configuration, have not been clarified [2]. In this work, we propose a code deformation procedure that transforms standard patches into a densely packed state. We further designed a cnot gate-scheduling scheme for syndrome measurements tailored to this densely packed configuration, and validated its effectiveness through logical error rate simulations.

The structure of this paper is as follows. First, Sec. II provides an overview of the fundamental concepts of the surface code as preliminary knowledge, followed by an explanation of the code deformation necessary to achieve dense packing. Section III discusses the upper limit of physical qubits that can be reduced in densely packed surface codes. We also propose a conceptual microarchitecture that employs dense packing for space-efficient fault-tolerant quantum computing. Section IV describes a code deformation procedure that transforms standard surface-code patches into a densely packed configuration. Section V provides cnot gate scheduling for stabilizer measurement circuits in densely packed surface code, which suppresses hook errors. Finally, Sec. VI presents simulation results of the logical error rate in the densely packed state. We found that densely packed surface code demonstrates logical error rates comparable to or lower than those of standard surface code.

II. BACKGROUND

This section explains the basic aspects of surface codes and how they can be deformed.

A. Surface codes

Reliable quantum computation requires protecting fragile quantum states from errors. This protection can be achieved by encoding logical qubits onto redundant physical qubits with quantum error-correcting (QEC) codes. Many QEC codes, including the surface codes, are constructed on the basis of the stabilizer formalism [28]. By performing syndrome measurements of stabilizer generators, errors can be detected without destroying the encoded logical information.

Among the various QEC codes, the surface code [8,11,17–20] stands out as a leading candidate due to its high threshold error rate [29] and compatibility with two-dimensional architectures requiring only nearest-neighbor interactions. As a result, many fault-tolerant quantum computing architectures [30–32] are based on the surface code.

Figure 1 shows a patch of the surface code. In this configuration, physical qubits are arranged on a two-dimensional lattice. The black dots denote physical data qubits, while the white dots represent the auxiliary qubits used for syndrome measurements, which we refer to as measurement qubits. The regions delineated by squares and triangles represent stabilizer generators. Here, dark-shaded areas correspond to X-type stabilizers and light-shaded areas correspond to Z-type stabilizers. Errors on data qubits are detected through syndrome measurements of the stabilizer generators. The red and blue lines are referred to as the X and Z boundaries, respectively. A sequence of $X(Z)$ gate operations on the physical qubits, along with $X(Z)$ boundaries, corresponds to a logical $X(Z)$ gate on the surface code. The code distance d of a surface code is defined as the minimum weight of a nontrivial logical operator.

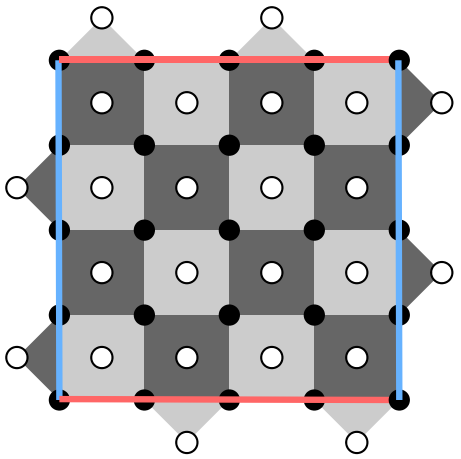


FIG. 1. A $[\![25, 1, 5]\!]$ surface-code patch. The black dots represent the data qubits of the patch, while the white dots denote the measurement qubits used for syndrome measurements. The dark-shaded regions and light-shaded regions represent the $X^{\otimes 4}(X^{\otimes 2})$ and $Z^{\otimes 4}(Z^{\otimes 2})$ stabilizer generators, respectively. Here, $X^{\otimes 4}$ and $Z^{\otimes 4}$ denote the square stabilizer generators acting on four data qubits, and $X^{\otimes 2}$ and $Z^{\otimes 2}$ denote the boundary stabilizer generators acting on two data qubits at the edges of the patch. The red lines on the patch denote X boundaries, and the blue lines denote Z boundaries.

Figure 2(a) shows the single X-type stabilizer generator excerpted, illustrating required two-qubit gates between

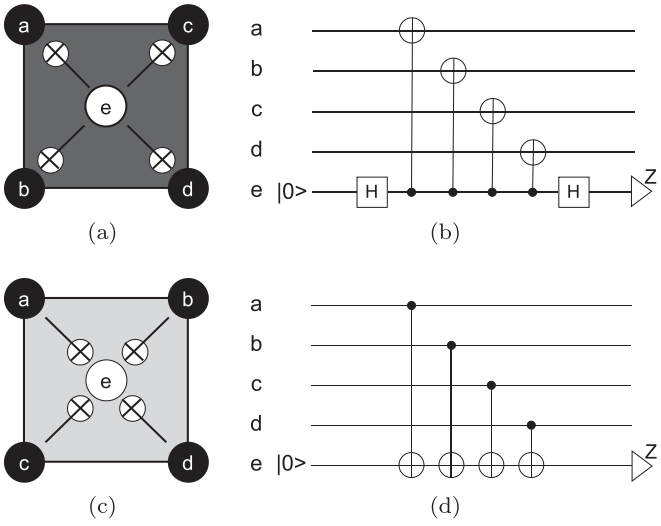


FIG. 2. (a) X-type stabilizer of a surface code, which corresponds to each dark-shaded region in Fig. 1. It consists of supports of the $X^{\otimes 4}$ stabilizer as black dots and a measurement qubit as a white dot in the same way as Fig. 1. Four lines with a $\oplus$ sign depicted in the figure correspond to commutative CX gates necessary for syndrome measurement. (b) A circuit implementation of syndrome measurement for a $X^{\otimes 4}$ stabilizer. (c) Z-type stabilizer of the surface code, which corresponds to each light-shaded region in Fig. 1. (d) A circuit implementation of syndrome measurement for a $Z^{\otimes 4}$ stabilizer.

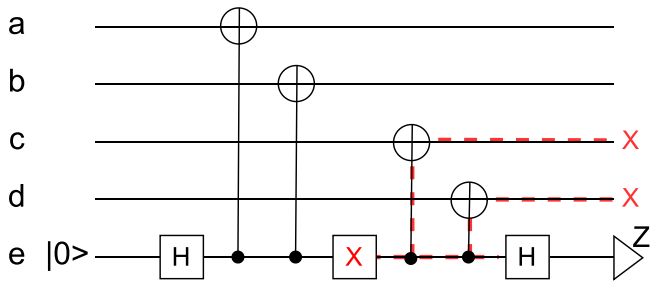


FIG. 3. An example of the hook error in a syndrome extraction circuit. An X error on the measurement qubit propagates to the two data qubits c and d.

data qubits and a measurement qubit for the syndrome measurement to perform the syndrome extraction. Figure 2(b) depicts the exact quantum circuits for the syndrome extraction. Figures 2(c) and 2(d) similarly describe the Z-type stabilizer. When an odd number of Z errors occur on this X-type stabilizer, the syndrome obtained from this stabilizer flips. Errors on data qubits are corrected using the outcomes of syndrome measurements of the stabilizer generators. However, these syndrome measurements themselves are subject to noise. To reliably perform error correction, it is therefore necessary to repeat the syndrome measurements multiple times [12]. A complete cycle in which all stabilizers are measured once is referred to as a round, and typically d rounds of measurements are performed where d is equal to the code distance. This ensures that errors, including measurement errors, can be corrected, thereby achieving protection with a distance d in both time and space.

In stabilizer syndrome measurement circuits, errors occurring on measurement qubits can propagate to multiple data qubits. Figure 3 illustrates an example where a single X error on the measurement qubit during the measurement of an $X^{\otimes 4}$ stabilizer propagates as X errors on two data qubits. Such errors are referred to as hook errors [19]. For Z-type stabilizers, an analogous argument holds with X and Z interchanged.

Figure 4 illustrates how hook errors in the patch extend the length of error chains. In this case, X errors occurring on the measurement qubits of the X-type stabilizer generators propagate to the data qubits, turning two physical errors into four physical errors. Because the errors align with the logical X operator in the patch, a logical X error can occur with just two physical errors, despite the original code distance being 5. Additionally, a Z error occurring on the measurement qubit of the Z-type stabilizer generator propagates to the data qubits, resulting in one physical error becoming two physical errors on the data qubits. However, the two propagated Z errors are orthogonal to the logical Z operator and do not increase the logical error rate of the patch.

Since hook errors may cause error propagations that increase the logical error rate, two-qubit gate scheduling must be determined with these propagations in mind [15,20,33,34]. The propagation direction of hook errors is determined by the scheduling of the two-qubit gates. By determining appropriate gate scheduling so that $X(Z)$-type hook errors occur perpendicular to the direction of the $X(Z)$ logical operators on the patch, the minimum number of physical errors required to

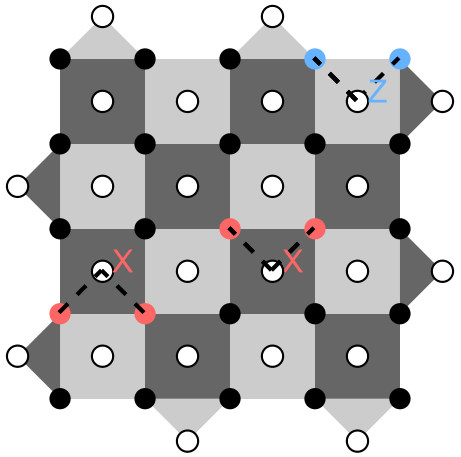


FIG. 4. As shown in Fig. 3, errors on measurement qubits propagate to two data qubits. The X-type hook errors propagate along the logical X operator, while the Z-type hook errors propagate perpendicular to the logical Z operator.

cause a logical error can be kept unchanged even when hook errors occur. Section V presents a detailed discussion of gate scheduling.

B. How to deform surface codes

The configuration of a code can be altered via code deformation [35]. Code deformation is a technique that modifies the generating set of measured stabilizers to transition the code from one state to another. In Fig. 5, code deformation is used to extend and contract a surface code patch. Figure 5(a) shows a single patch of the surface code together with physical qubits, which are initialized in the $|0\rangle$ state. In Fig. 5(b), stabilizer syndrome measurements are performed including the physical qubits initialized in (a). After d rounds of measurements, the patch is effectively expanded. Also, the patch can be reduced by measuring its physical qubits. For instance, in Fig. 5(c), measuring the data qubits along the X boundary in the Z basis shortens the length of the Z boundary. In Sec. IV, we demonstrate that this method enables multiple surface code patches to be transformed into a more densely packed configuration and restored.

III. REDUCING THE SPACE OVERHEAD OF SURFACE CODE

This section discusses the space overhead of surface codes that can be reduced by dense packing and a conceptual microarchitecture utilizing dense packing.

A. Reducing spatial overhead via dense packing

In the previous study [2], it has been proposed to deform multiple surface codes into a single high-density code as illustrated in Fig. 6. In this configuration, the logical operators X and Z are represented as red and blue lines, respectively. Although multiple surface codes are fused into a single state, the original logical states are preserved. This configuration allows the logical state to be maintained with fewer physical qubits than in the standard surface-code layout. We refer to

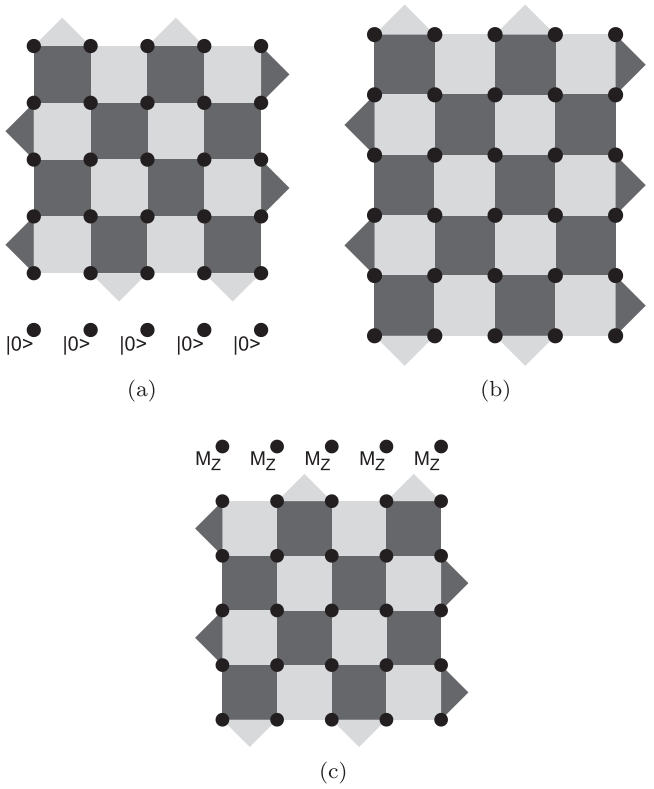


FIG. 5. Code deformation procedures to extend and contract a surface code patch. (a) A surface code patch and physical qubits initialized to the $|0\rangle$ state, (b) by introducing additional stabilizers to the surface code patch, it can be extended after d rounds of syndrome measurement, and (c) By measuring the data qubits on the surface code in the Z basis, it can become shorter.

this technique as dense packing. In contrast to this technique, in this paper, we refer to the conventional, nondensely packed surface-code patch as the *Stand-alone surface-code patch*, to emphasize that it functions as an independent single logical qubit patch with a separated layout.

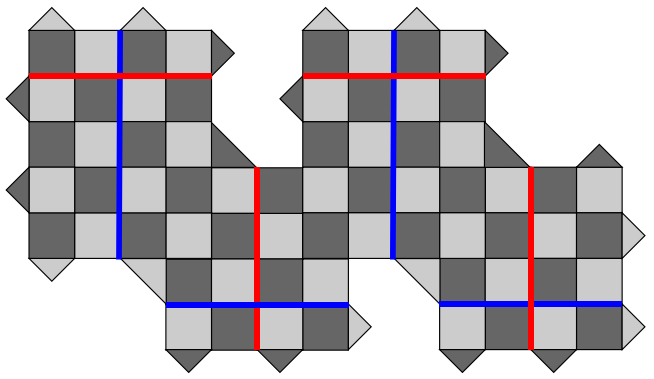


FIG. 6. An example of densely packed [2] surface-code code words. Four logical qubits are packed in this case, thereby realizing a more efficient encoding that requires a reduced area, resulting in lower physical-qubit overhead. The red and blue lines correspond to the logical X and Z operators of the logical qubits, respectively.

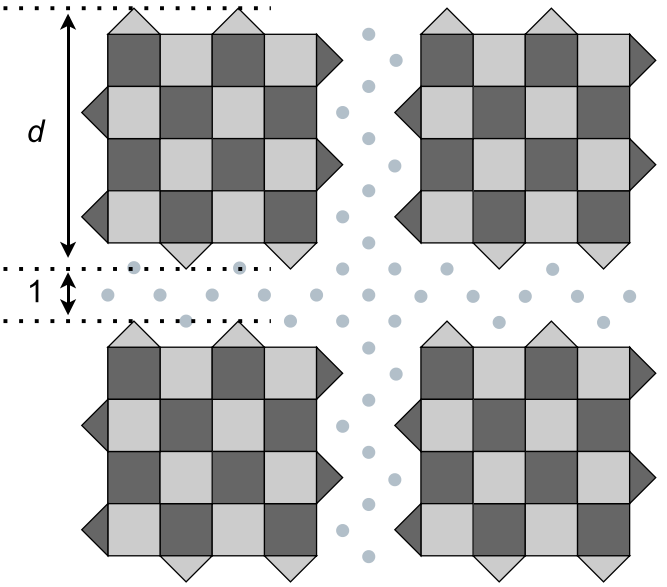


FIG. 7. A layout of stand-alone surface-code patches without dense packing. The gray dots represent physical qubits. These physical qubits are not accessed for quantum computation.

Although the prior study has suggested that dense packing reduces the number of qubits used to approximately three-fourths of the original count, the details of this reduction were not explicitly clarified. This also applies to the allocation strategy of densely packed surface codes. In addition to the horizontal fusion forming a row as shown in Fig. 6, vertical fusion can also be considered. In this section, we examine the conditions under which the allocation strategy of dense packing becomes effective and show the reduction rate of the space overhead of surface codes under these conditions.

In the previous study, the patch lengths of the upper and lower surface codes were different. However, in this discussion, it is assumed that the patch lengths are the same in both the X and Z directions, and that all surface codes have identical patch lengths. It is assumed that the patch length of the surface codes is an odd number.

As a measure of the physical qubit requirement per logical qubit, we use the space occupied by a single logical qubit rather than the number of physical qubits required for its encoding. This is because, when surface code patches are arranged on a square lattice, there arise physical qubits that cannot be accessed for computation, even though they are not used as data qubits or measurement qubits on the surface code. This is shown in Fig. 7. Therefore, in the following discussion, we evaluate the space overhead of the surface code required per logical qubit.

When stand-alone surface-code patches are arranged, their layout appears as shown in Fig. 7, where n_h codes are placed vertically, and n_w codes are placed horizontally. The total area occupied by these $n_h n_w$ surface codes is given by

$$\{n_h d + (n_h - 1)\}\{n_w d + (n_w - 1)\} = n_h n_w (d+1)^2 - (n_h + n_w)(d+1) + 1. \quad (1)$$

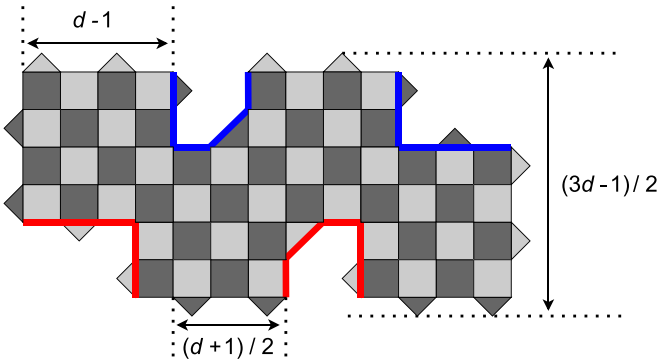


FIG. 8. Densely packed surface codes. The individual code distances are kept as close to d as possible, and the distances of logical operators across multiple codes, as shown by the red and blue lines, are kept d or more. The red and blue lines correspond to the X-type and Z-type multiqubit logical operators, respectively.

Consequently, the area occupied per logical qubit is

$$(d+1)^2 - \frac{(n_h+n_w)(d+1)}{n_h n_w} + \frac{1}{n_h n_w}. \tag{2}$$

On the other hand, when the surface codes are arranged with dense packing while maintaining the same patch length, their layout changes to that shown in Fig. 8. In this case, the total area occupied by n surface codes is

$$\begin{aligned}&\frac{3d-1}{2}\left\{(d-1)+(n-1)\frac{d+1}{2}+1\right\}\\ &\quad=\frac{(3d-1)(d+1)n}{4}+\frac{(3d-1)(d-1)}{4}.\end{aligned} \tag{3}$$

Assuming $n \gg d^2$, the area occupied per logical qubit becomes

$$\begin{aligned}&\frac{(3d-1)(d+1)}{4}+\frac{(3d-1)(d-1)}{4n}\\ &\quad\approx\frac{(3d-1)(d+1)}{4}.\end{aligned} \tag{4}$$

By comparing this result to the first case, it can be confirmed that as n becomes significantly larger than d^2, the area occupied per logical qubit asymptotically reduces to 3/4 of the original value.

By extending the dense packing arrangement shown in Fig. 8 vertically, it is conceivable to further increase the number of surface codes packed densely. For example, the configuration illustrated in Fig. 9 can be considered. If the arrangement in Fig. 9 consists of n_w surface codes per row and n_h such rows stacked vertically, the total area occupied by the densely packed surface codes is given by

$$\begin{aligned}&\left\{\frac{3d-3}{2}(n_h-1)+\frac{3d-1}{2}+1\right\}\\ &\quad\times\left\{(d-1)+\frac{d+1}{2}(n_w-1)+1\right\}\end{aligned} \tag{5}$$

$$\begin{aligned}&=\frac{1}{4}\{(3d-3)(d+1)n_h n_w+(3d-3)(d-1)n_h\\ &\quad+4(d+1)n_w+4(d-1)\}.\end{aligned} \tag{6}$$

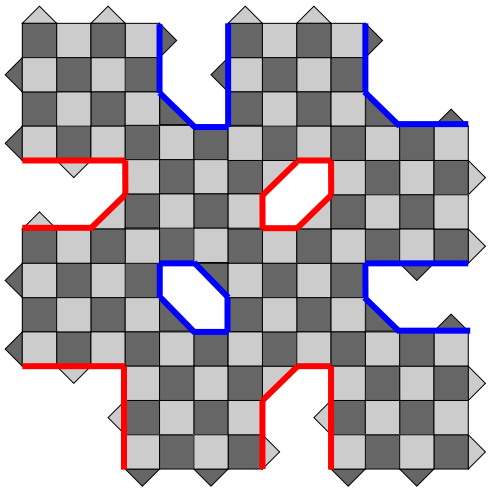


FIG. 9. Example of Dense packing extension. In addition to using dense packing in the horizontal direction, it is possible to use it in the vertical direction as well. The individual code distances are kept as close to d as possible, and the distances of logical operators across multiple codes, as shown by the red and blue lines, are kept d or more. The red and blue lines correspond to the X-type and Z-type multiqubit logical operators, respectively.

The area occupied per logical qubit is then expressed as

$$\begin{aligned}&\frac{1}{4}\left\{(3d-3)(d+1)+\frac{(3d-3)(d-1)}{n_w}\right.\\ &\quad\left.+\frac{4(d+1)}{n_h}+\frac{4(d-1)}{n_h n_w}\right\}.\end{aligned} \tag{7}$$

As n_w becomes larger, the area used relative to that of a standalone surface code asymptotically approaches 3/4. Therefore, it is preferable to make n_w as large as possible. This preference arises because vertical dense packing introduces defects that span multiple surface-code patches, creating logical structures whose code distance must be preserved.

When used as quantum storage, accessing logical qubits located at the center of the structure becomes challenging. Furthermore, due to defects in the densely packed surface code, gate scheduling is expected to be more complex than the case with Fig. 8. The exact procedure will be described later, only for the linear configuration in Sec. V. For practical quantum computing, it may be more efficient to maintain multiple instances of the configuration shown in Fig. 8. We also note that, although denser configurations based on dense packing than that shown in Fig. 9 may exist, we focus on the approach using Fig. 8 for the same practical considerations discussed above.

B. A conceptual microarchitecture with hierarchical memory

In large-scale fault-tolerant quantum computing, many logical qubits act as long-term memory rather than active computation units. Treating all patches equally leads to inefficient use of spatial resources. By employing densely packed surface codes as a memory layer, we can realize a hierarchical quantum memory that trades off access time for spatial efficiency.

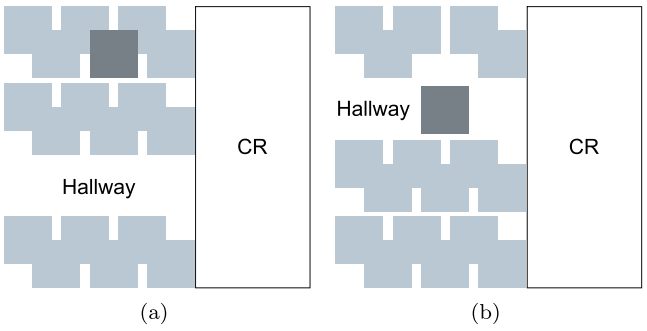


FIG. 10. A layout for a microarchitecture with hierarchical memory. Densely packed surface codes are used to provide space-efficient storage for infrequently accessed logical qubits. (a) The logical patch that the computational region (CR) needs to access is depicted as a black square, but other rows are obstructing it. (b) By moving the second row downward, the logical patch faces the hallway, allowing access from the computational region.

Here, we propose one possible conceptual design of a microarchitecture that introduces a memory hierarchy by employing both dense-packed and stand-alone surface codes. This microarchitecture is divided into two regions: a computational region and a memory region, inspired by the Load/Store architecture [27]. The computational region adopts the stand-alone configuration to achieve highly parallel time-efficient computation, while the memory region utilizes multiple rows of densely packed surface codes to provide space-efficient storage for patches that are not frequently accessed.

Figure 10 shows the layout of the microarchitecture. The light-gray areas represent densely packed surface codes and a stand-alone surface-code patch in the memory region. In Fig. 10(a), densely packed rows are arranged, and also a row of empty qubits—referred to as a *hallway*—is provided to enable selective access to specific logical qubits from the computational region.

When attempting to access the logical information stored in a row that is not facing the hallway (e.g., the topmost row, illustrated in dark gray in Fig. 10), the other row (the second row in this example) obstructs direct access to the first row's logical information. In such a case, one can shift the entire second row downward as shown in Fig. 10(b) with $2d$ rounds plus two parallel physical qubits measurement steps. The code deformation procedure for the movement of the entire row of densely packed surface codes is provided in Appendix B. After this shift, the first row faces the hallway, and the desired patch is separated from its neighbors. This movement of the hallway enables access to any logical information stored in the memory region. The procedure for separating a single logical qubit from the densely packed configuration is given later in Sec. IV. Note that the computational region of this microarchitecture is compatible with existing surface-code patch layouts (so-called *floorplans*) without dense packing, such as those in Refs. [36–40].

IV. TRANSFORMING SURFACE CODES INTO DENSELY PACKED STATE

In this section, we demonstrate that the dense packing configuration can be achieved through the use of code deformation. The code deformation can be utilized for logical gate [35], but here we perform the deformation only for fusing and splitting code words.

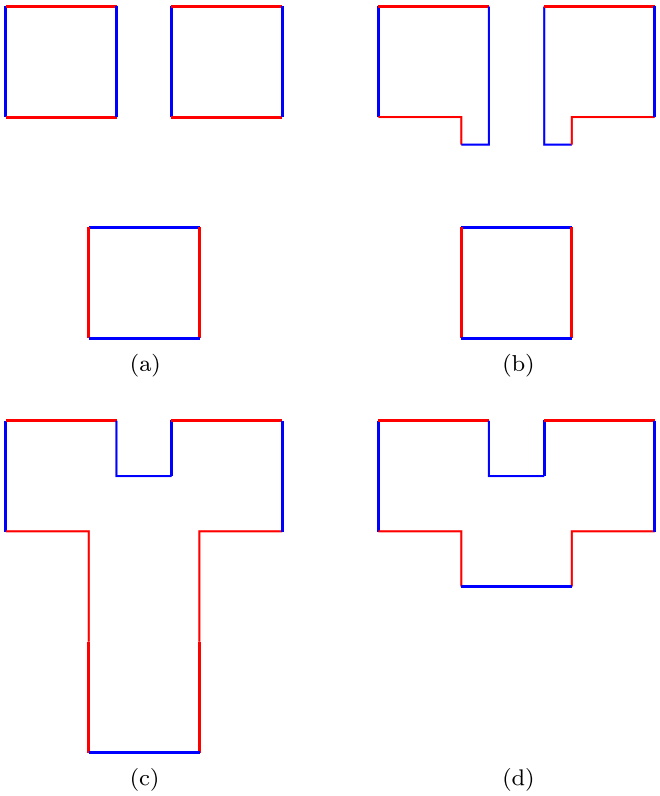


FIG. 11. The squares in the figure are schematic representations of surface-code patches, illustrating the process by which three surface codes transform into a single dense packing state. The red and blue lines denote the X and Z boundaries, respectively. (a) The initial configuration of the surface code patches, which will be densely packed, (b) among the three patches, the top two are deformed. Consequently, the upper patches and the lower single patch face the same boundary, (c) the three codewords have been fused to a single contiguous region, while preserving the original logical states of all three, and (d) finally, by reducing the code distance, the dense packing configuration is achieved.

The transformations required for dense packing using code deformation are described below. Figure 11 illustrates the condensed procedure for deforming the surface code; for a more detailed description of the deformation steps, refer to Appendix A. In Fig. 11, the squares represent patches of the surface code, and this figure serves as a simplified picture of Fig. 1. The blue lines denote Z boundaries, while the red lines denote X boundaries. The figure schematically illustrates how surface codes are deformed through the use of code deformation.

In Fig. 11, transforming the patches of the surface code from (a) to (b) and from (b) to (c) each requires d rounds. In contrast, the transformation from (c) to (d) can be achieved solely by physical qubit measurement. Therefore, the total time to transform the surface-code patches from their stand-alone configuration to the densely packed configuration is $2d$ rounds plus the time required for physical qubit measurements. To separate a given patch of the surface code from the

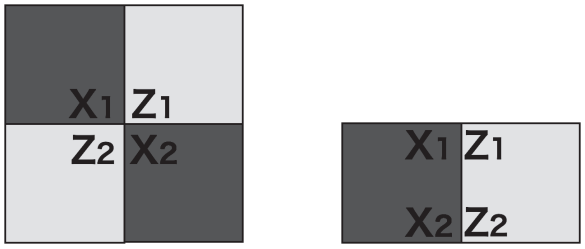


FIG. 12. The constraints for gate scheduling in surface codes. (a) Constraint 1. When applying stabilizers on the surface code using CNOT gates, it is not allowed to apply cnot gates to the same qubit simultaneously. Therefore, the timing of the cnot gates acting on X_1, X_2, Z_1, and Z_2 must not overlap. (b) Constraint 2. The order in which CNOT gates are applied must satisfy either $X_1 < Z_1$ and $X_2 < Z_2$, or $X_1 > Z_1$ and $X_2 > Z_2$.

densely packed state, the above steps must be performed in reverse order. For example, to extract the lower patch of the surface code from (d), the deformation from (d) back to (c) requires d rounds, and the transition from (c) to (b) is performed by physical-qubit measurements. Therefore, the time required to extract a specific surface code from the densely packed state is d rounds plus the time for physical qubit measurements.

V. HOOK-ERROR-AVOIDING GATE SCHEDULING

Surface codes require both X-type and Z-type stabilizer measurements. When performing these measurements, it is necessary to determine appropriate cnot gate scheduling within the stabilizer measurement circuit.

Gate scheduling for stabilizer measurements in the surface code must satisfy two constraints to reproduce correct stabilizer measurements. The first constraint is that when applying gates to data qubits shared by multiple stabilizers, only a single gate can be applied at the same time, as illustrated in Fig. 12(a).

The second constraint requires that the sequence of CNOT gate operations for a given stabilizer measurement must not disturb the outcomes of adjacent stabilizer measurements. To satisfy this constraint, it is sufficient to ensure that for the two data qubits shared between neighboring stabilizer measurement circuits, the cnot gates from one stabilizer measurement circuit always precede those from the other. As illustrated in Fig. 12(b), let the time steps at which each stabilizer measurement circuit applies CNOT gates to the data qubits be denoted by X_1, X_2, Z_1, Z_2. A valid scheduling requires that either $X_1 < Z_1$ and $X_2 < Z_2$, or $X_1 > Z_1$ and $X_2 > Z_2$.

Furthermore, considering the suppression of error propagation caused by hook errors in directions parallel to the logical axis, one possible CNOT gate scheduling is as shown in Fig. 13.

When code words are densely packed, multiple logical qubits are mapped onto a single contiguous surface-code patch. As a result, the same-basis logical observables of two connected logical qubits are arranged perpendicularly, necessitating modification of the gate scheduling.

We propose the scheduling shown in Fig. 14 as a method that fulfills both of the above constraints and largely mitigates the impact of hook errors. In this gate scheduling, hook errors

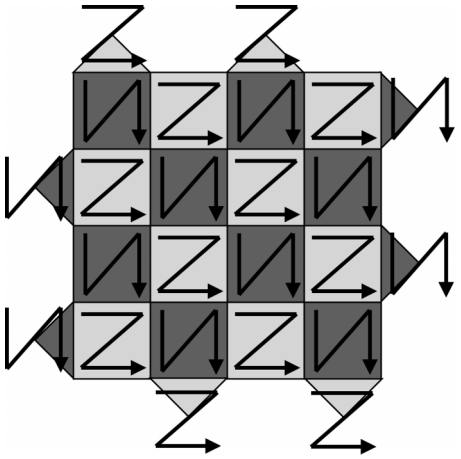


FIG. 13. Hook-error-avoiding gate scheduling for stand-alone surface code patch. When applying two-qubit gates between data qubits at the lattice vertices and the measurement qubits, the gates must be executed in the order indicated by the arrows.

are not entirely prevented from propagating in the logical direction. However, the regions where logical-direction error propagation occurs are limited to a small subset of stabilizers, and their overall impact on the logical error rate is considered to be small.

This gate scheduling consists of five time steps for applying cnot gates, resulting in a longer stabilizer measurement duration compared to stand-alone surface-code gate scheduling. While this might suggest an increase in the error rate compared to the stand-alone surface code, in practice, the error rate remains effectively suppressed even in a densely packed configuration. In Sec. VI, we verify this through simulations employing this gate scheduling in a densely packed arrangement.

VI. ERROR RATES SIMULATION OF DENSELY PACKED SURFACE CODE

To estimate the circuit-level logical error rate of the densely packed surface codes, we constructed quantum circuits using

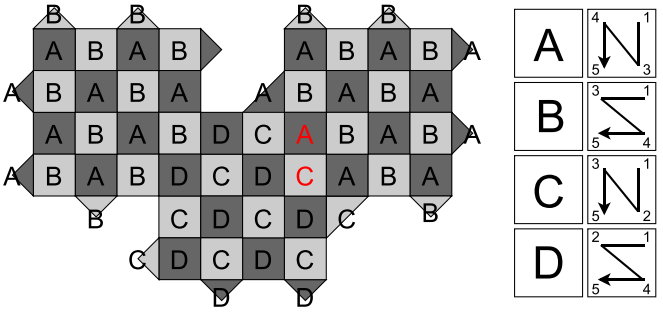


FIG. 14. Gate scheduling for densely packed surface codes. A, B, C, and D represent the order in which the gates are applied. By adjusting the gate scheduling, the propagation of hook errors in the logical direction is suppressed. However, in the area indicated by red text, hook errors still propagate in the logical direction. These stabilizers represent only a small portion of the surface code and are therefore not expected to have a significant impact.

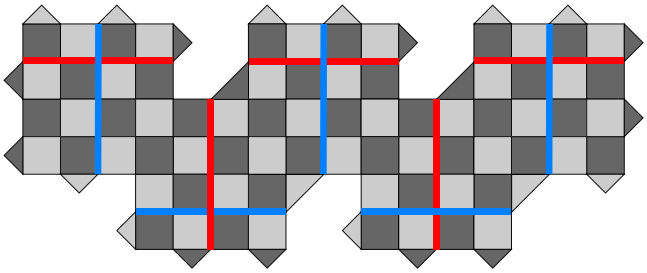


FIG. 15. The densely packed patch encoding 5 logical qubits that is used in the error correction simulation. The simulation focused on evaluating the error rate of only the central logical qubit. The red and blue lines correspond to the logical X and Z operators of the logical qubits, respectively.

the python library stim [41], and performed syndrome decoding using pymatching [42]. The codes used for the simulation are available in Ref. [43].

We performed three types of simulations. The first simulation considered the densely packed surface-code patches that adopt the gate scheduling shown in Fig. 14. As illustrated in Fig. 15, we constructed a densely packed surface code of five code words and evaluated the logical error rates of all five logical qubits. We refer to this densely packed surface code as hook-avoiding dense. At the end of the error correction procedure, all physical qubits were destructively measured to verify whether logical errors occurred along the logical operators indicated by the red and blue lines.

The second simulation also examined the densely packed state, but employing a gate scheduling that does not account for hook errors. We refer to this densely packed surface code as hook-prone dense. Same as the first case, a five-patch densely packed surface code was constructed as shown in Fig. 15. In this surface code, the scheduling of the X-type stabilizers was fixed to pattern B in Fig. 14, while the scheduling of the Z-type stabilizers was fixed to pattern A. Both schedules (A and B) were adapted to four time steps. Under this scheduling, the two code words at the bottom of Fig. 15 are protected from hook errors, whereas the three code words at the top remain vulnerable to them. The third simulation evaluated the logical error rates of the stand-alone surface code, not densely packed. In this case, the scheduling of the X-type stabilizers was fixed to pattern A, and that of Z-type stabilizers was fixed to pattern B, with both schedulings adapted to four time steps. These schedulings correspond to a conventional scheme designed to suppress the impact of hook errors.

This is achieved by simulating the logical error rate corresponding to X errors on the logical $|0\rangle$ state and Z errors on the logical $|+\rangle$ state. We evaluated both the logical X error rate and the logical Z error rate because there exists asymmetry in the susceptibility to logical X errors and Z errors for each code word on the densely packed surface code. Focusing on the orientation of the logical $Z(X)$ operators of the central code word in Fig. 15, we can see that graphical asymmetry can lead to asymmetry between the logical X error rate and the logical Z error rate. When extending the densely packed surface code in Fig. 15 further, the code words in the upper region will exhibit the same asymmetry as the central code word, while the code words in the bottom region will exhibit reversed $X(Z)$ error rate asymmetry.

The maximum number of shots is set to 10^8, and the maximum number of errors is limited to 10^4.

The noise model used in the simulation assumes that two-qubit gates, measurement, and initialization operations are the noisy processes. The details are as follows:

(1) Each CNOT gate is followed by two-qubit depolarizing noise with a probability p.

(2) Each measurement operation is followed by a result flip with a probability p and single-qubit depolarizing noise with a probability p.

(3) Each initialization operation is followed by a result flip with a probability p.

(4) Each single-qubit gate is followed by single-qubit depolarizing noise with a probability $p/10$.

(5) Each idle qubit location (where no gate operation occurs) is followed by single-qubit depolarizing noise with a probability $p/10$ during each operation.

Single-qubit depolarizing noise with a probability p means that during each single-qubit gate operation, an X, Y, Z error occurs on the qubit with a probability of $p/3$ each, while no error occurs with a probability of $1-p$. Two-qubit depolarizing noise with a probability p means that during each two-qubit gate operation, one of the $\{I\otimes X, I\otimes Y, I\otimes Z, X\otimes I, X\otimes X, X\otimes Y, X\otimes Z, Y\otimes I, Y\otimes X, Y\otimes Y, Y\otimes Z, Z\otimes I, Z\otimes X, Z\otimes Y, Z\otimes Z\}$ errors occurs with a probability of $p/15$ each, while no error occurs with a probability of $1-p$.

The simulation results are shown in Fig. 16. Although we evaluated logical error rates of all logical qubits, we present simulation results only for the central and the second-from-the-left logical qubits. This is because the logical qubits can be classified into two groups: leftmost, central, and rightmost logical qubits exhibit the same scaling behavior of the logical error rate, while the second-from-the-left and second-from-the-right logical qubits share another same scaling behavior.

We first examine the logical error rate of the central logical qubit. We compare the logical error rates of two types of densely packed surface codes. We found that hook-avoiding dense yielded lower logical error rates than hook-prone dense. Then, we compare the logical error rates of the hook-avoiding dense and the stand-alone surface code. We observed that the stand-alone surface code patches generally yielded lower logical error rates when the physical error rate is around 10^{-2}. However, as the physical error rate decreases, the hook-avoiding dense exhibited lower logical error rates than the stand-alone surface code.

In the hook-avoiding dense, the number of times slices of CNOT gates increases from four to five compared to the stand-alone. This increases the likelihood of physical errors on idle qubits. However, for code distances $d \geqslant 7$, the logical error rates of the hook-avoiding dense become lower than that of the stand-alone patch as the physical error rate decreases. This improvement can be attributed to the fact that, in the hook-avoiding dense configuration, multiple patches share common regions where logical errors are less likely to occur. In stand-alone surface-code patches, a logical error can occur with as few as $(d+1)/2$ physical errors. In contrast, in the hook-avoiding dense, a logical error that spans

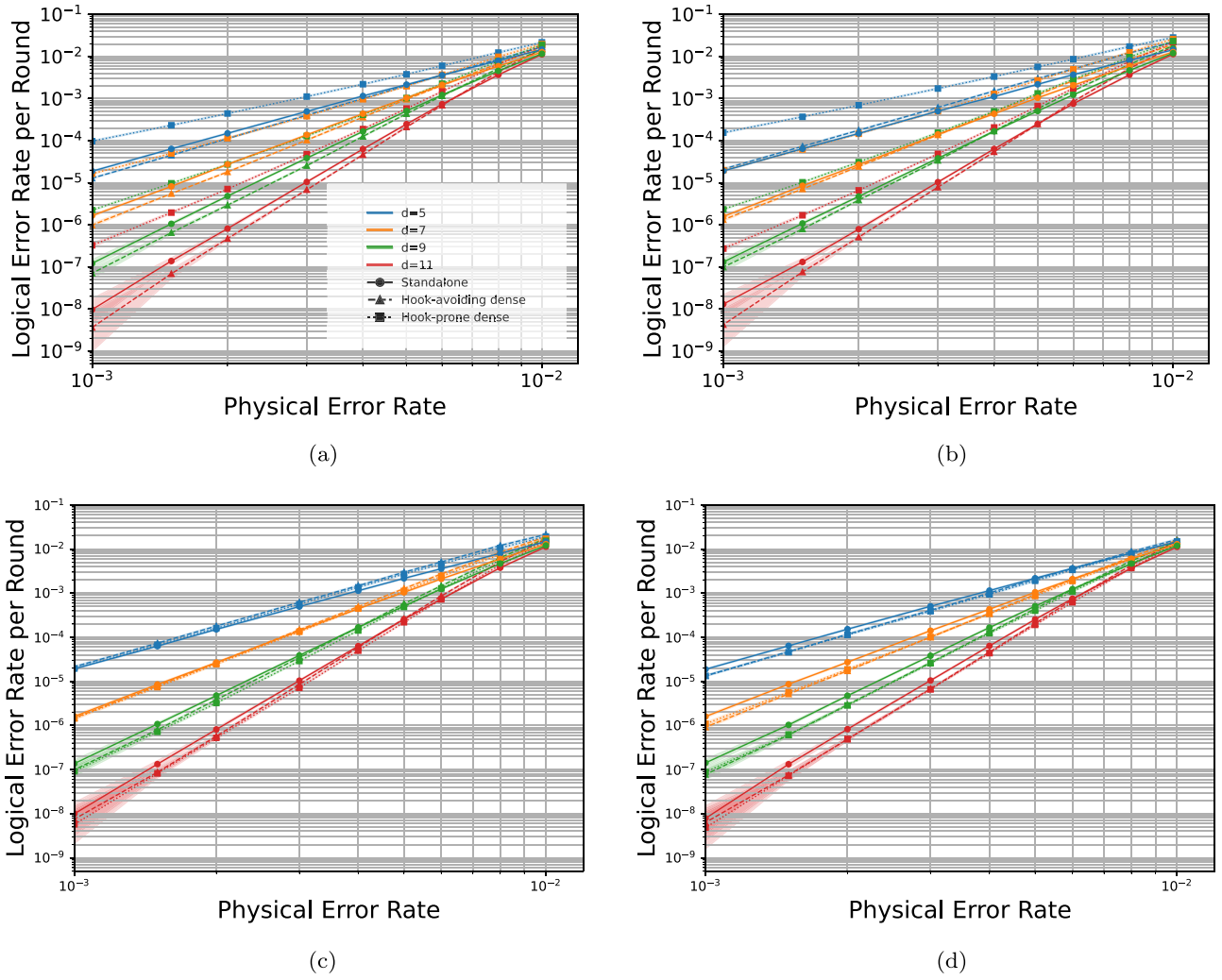


FIG. 16. Simulation results of logical error rates of surface codes with different configurations. The horizontal axis shows the physical qubit error rate, and the vertical axis shows the logical error rate per round. Error correction was performed for $3d$ rounds. The solid, dashed, and dotted lines correspond to the stand-alone surface code, hook-avoiding dense, and hook-prone dense, respectively. Hook-avoiding dense adopts the gate scheduling in Fig. 14, while hook-prone dense uses a scheduling that does not account for hook errors. The lightly shaded region indicates where the Bayes factor (likelihood ratio) reaches 1000. (a) Logical X error rate per round of central logical qubit, (b) logical Z error rate per round of central logical qubit, (c) logical X error rate per round of second from the left logical qubit, and (d) logical Z error rate per round of second from the left logical qubit.

across the fused regions requires more than $(d+1)/2$ physical errors. Since the regions where logical errors can occur by only $(d+1)/2$, physical errors are limited to nonfused parts.

On the other hand, the higher logical error rate observed in the hook-prone dense is likely due to the increased susceptibility to hook errors. The number of times CNOT gates are applied for stabilizer measurements in this gate scheduling is the same as that in the stand-alone patch, and, the same as hook-avoiding dense, multiple patches share common regions. However, the logical error rates increase because hook errors propagate in the same direction as the logical operators.

Next, we consider the logical error rate of the second-from-the-left logical qubit. For this qubit, the relationship between the logical error rates of the stand-alone surface code and the hook-avoiding dense configuration is consistent with that observed for the central qubit. In contrast, we found that the logical error rate of the hook-prone dense configuration exhibits behavior very similar to that of the hook-avoiding dense configuration. This behavior can be attributed to the orientation of the logical $X(Z)$ operators of the second-from-the-left qubit, which are rotated by 90° relative to those of the central qubit. As a result, hook errors no longer propagate parallel to the logical operators, suppressing their contribution to logical failures.

However, in cases where hook errors influence the logical error rate, such as for the central logical qubit, the degradation of the logical error rate in the hook-prone dense configuration becomes pronounced. Therefore, when considering the overall logical error performance of the densely packed state, it is preferable to adopt the hook-avoiding dense gate scheduling.

Finally, in a densely packed configuration, correlated logical errors between multiple logical qubits may occur.

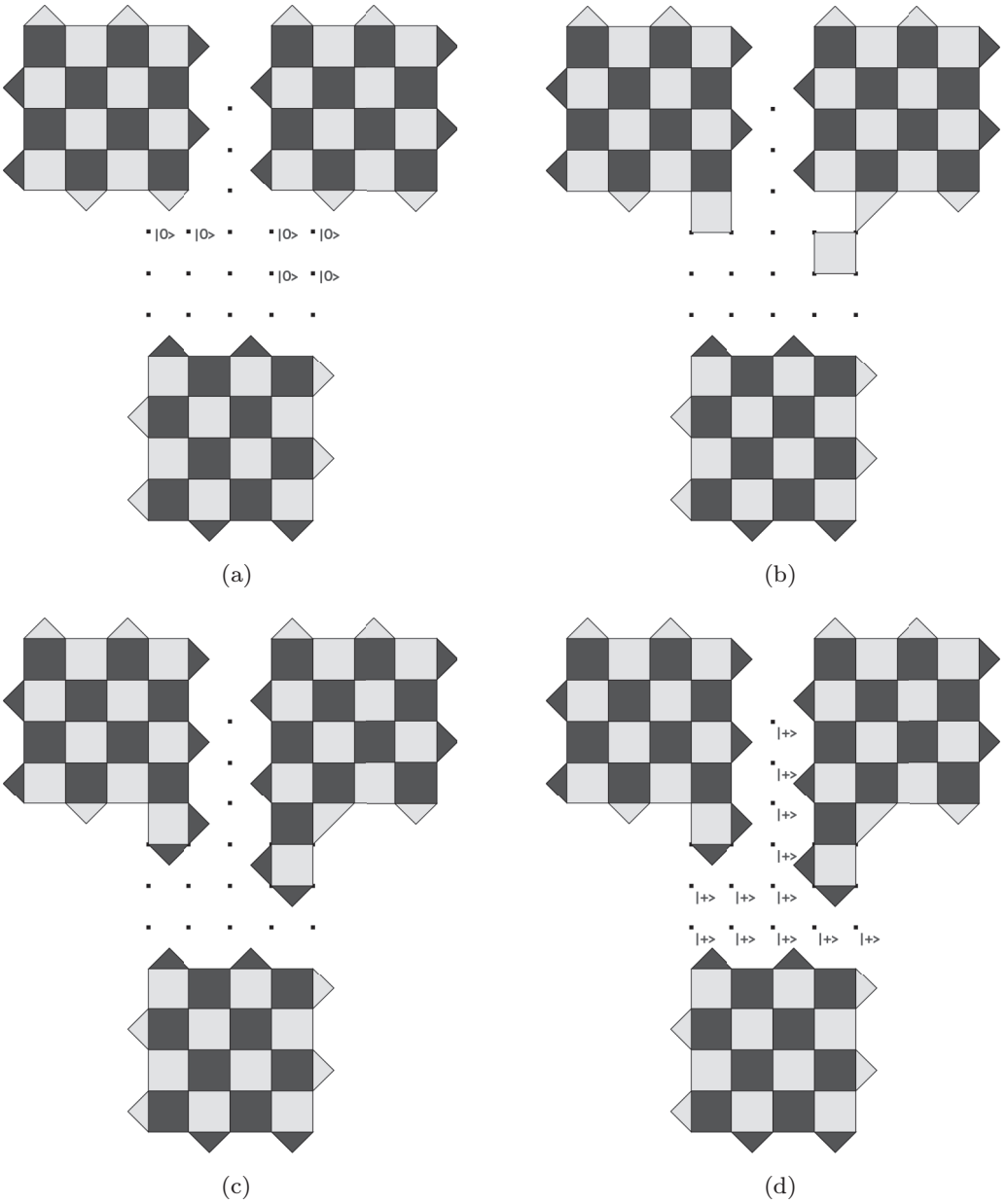


FIG. 17. Code deformation procedure for dense packing of the surface code. (a) The initial states for performing dense packing. The black dots represent physical qubits. These three standalone surface code patches subsequently undergo deformations to achieve the denser configuration, (b) the stabilizer generators associated with new data qubits which are in $|0\rangle$ state in (a) are determined without performing any measurements, (c) the upper two patches have been deformed, and the Z boundaries of these two patches are adjacent to the Z boundary of the lower patch, (d) two deformed surface code patches and a standalone surface code patch. Physical qubits are in $|+\rangle$ states, (e) the newly added dark shaded squares represent X-type stabilizers, which can be defined without stabilizer measurements, (f) the three patches have been fused together, (g) the lower part has become shorter from the (f) state and the three patches are in a densely packed state.

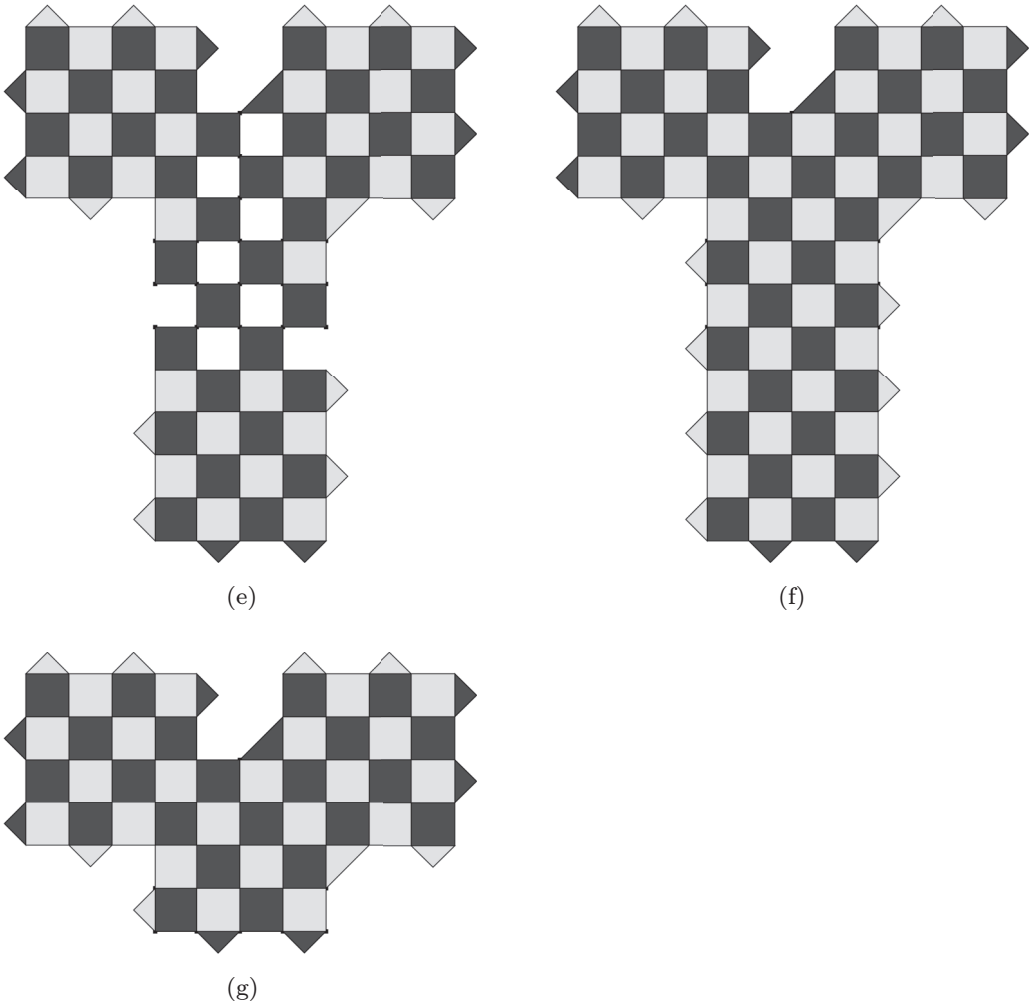
However, since the logical error rates of all logical qubits in the hook-avoiding dense configuration are comparable to or lower than those of the stand-alone surface code, we expect that any such correlations do not significantly degrade the overall logical error performance.

In this work, we present the logical error rates of the central and second-from-the-left logical qubits. The logical error rates of all logical qubits in the densely packed configuration can be evaluated using the code provided in Ref. [43].

This simulation evaluated the logical error rates only for code distances of 5, 7, 9, and 11. However, based on these results, it is expected that the hook-avoiding dense surface code will exhibit logical error rates comparable to or lower than the stand-alone surface code, even with larger code distances.

VII. CONCLUSION

In this paper, we propose an implementation method for dense packing of surface-code patches. It was confirmed that



(e) (f) (g)

FIG. 17. (*Continued.*)

the area required for densely packed surface-code code words asymptotically becomes three-fourths of that of stand-alone patches. We demonstrated that dense packing from stand-alone patches can be achieved through code deformation with $2d$ rounds plus the time required for physical qubits measurements, while extracting a logical qubit requires d rounds plus the time for physical qubits measurements.

The surface code in a densely packed state employs a gate scheduling different from the standard surface code. To minimize error propagation along the logical operator direction caused by hook errors, we propose to employ four types of gate scheduling for each stabilizer generator. In this approach, there exists one point per logical qubit where a hook error can propagate parallel along the logical operator direction. However, the region of the densely packed surface code affected by this error propagation is limited, and it is expected to have little impact on the logical error rate. This observation was verified through simulations, showing that even in a densely packed state, increasing the code distance and reducing the physical error rate result in a logical error rate that is comparable to or lower than that of the stand-alone surface code. Multiple logical qubits share part of the patch regions that would otherwise be independently assigned to each of them. In a stand-alone surface code, a logical error can occur with $(d+1)/2$ physical qubit errors. However, since densely packed surface code shares a patch region, the minimum number of physical qubit errors required to induce a logical error in that part increases. We believe this fact contributes to reducing the logical error rate of densely packed surface code. As the code distance increases and the physical error rate decreases, the logical error rate of densely packed surface code falls below the logical error rate of stand-alone surface code, and we believe this is due to the influence of this factor.

A practically important direction is the evaluation of the microarchitecture with a hierarchical memory approach. Our microarchitecture has the potential to reduce the spatiotemporal overhead of fault-tolerant quantum computing by employing space-efficient dense packing in the memory region and time-efficient stand-alone surface codes in the computational region. Evaluating the performance and its compatibility with various surface-code computing layouts [16,27,44] remains an important topic for future work.

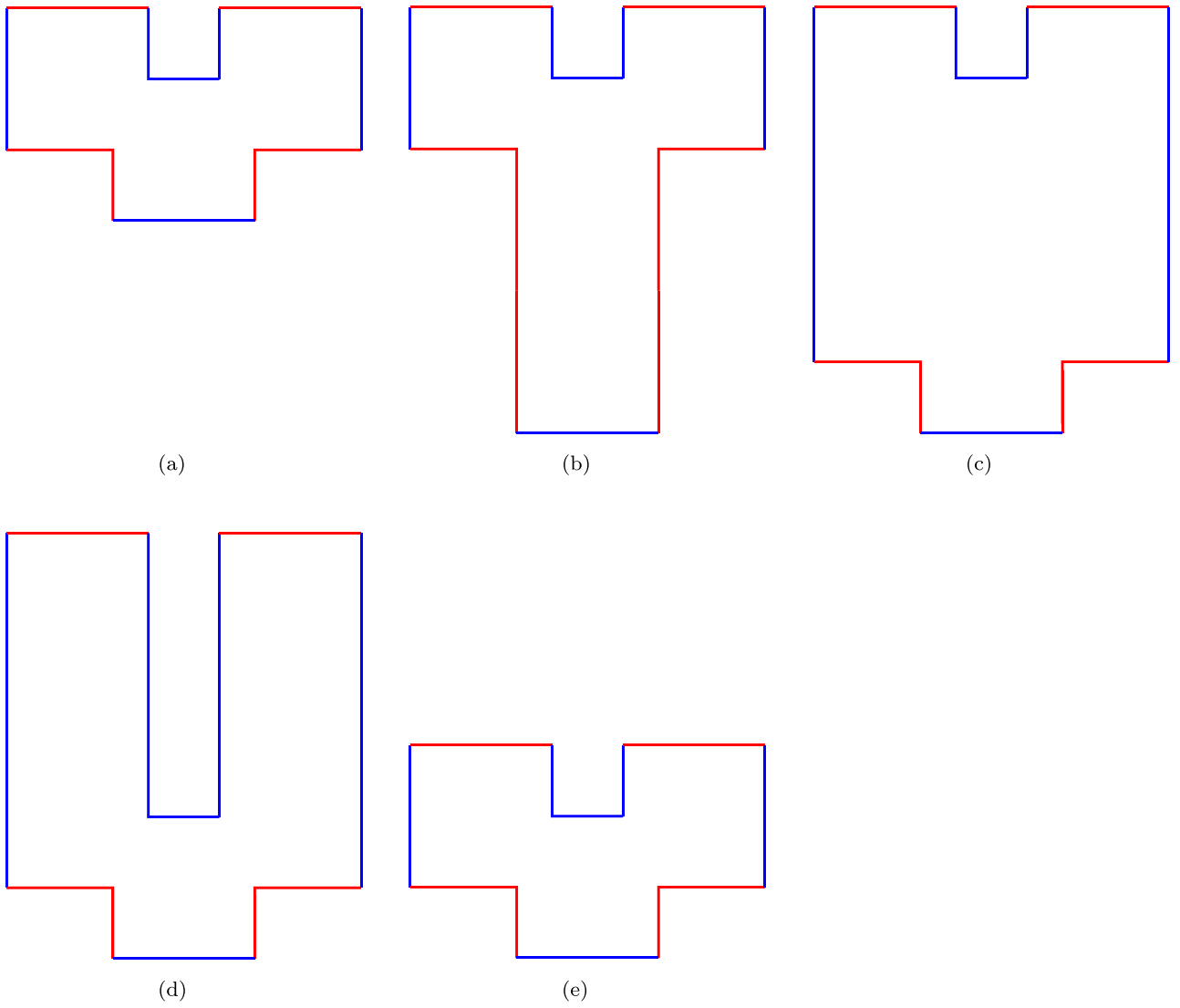


FIG. 18. Row-shifting movement in the densely packed surface code. (a) Densely packed surface code encoding three logical qubits. The space below the codes is the physical qubits used as a hallway, (b) the lower part of the densely packed surface code is extended downward, (c) the upper region is expanded downward, (d) the upper Z boundary has moved closer to the lower region, (e) the two protruding parts at the top have been reduced downward. The densely packed surface codes have finished shifting downward.

ACKNOWLEDGMENTS

This work was supported by JST Moonshot R&D Grant No. JPMJMS226C and the New Energy and Industrial Technology Development Organization (NEDO) project JPNP23003. S.N. has been supported by JSPS Overseas Research Fellowships.

DATA AVAILABILITY

The data that support the findings of this article are openly available [43].

APPENDIX A: SURFACE-CODE DEFORMATIONS FOR THE DENSE PACKING AT THE PHYSICAL QUBIT LEVEL

We describe the exact deformation procedures from the standard surface-code patches to the densely packed state at the physical qubit level. These deformations correspond to the condensed procedure shown in Fig. 11.

First, we prepare multiple stand-alone surface-code patches and physical $|0\rangle$ states as shown in Fig. 17(a). Without applying any physical operation, new stabilizer generators adjacent to the two upper patches can be defined as shown in Fig. 17(b). The eigenvalues of these new stabilizer generators are already determined because the newly added qubits are in $|0\rangle$ states. Then, we perform X and Z stabilizer measurements on the patches and the new stabilizer generators depicted in Fig. 17(c). After d rounds of this type of stabilizer measurement, the three patches stabilize in the state shown in Fig. 17(c). As a result, the Z boundaries of the upper two patches face the Z boundary of the lower patch. Simultaneously with the completion of the deformations of the upper two patches, we initialize the physical qubits in $|+\rangle$ states as shown in Fig. 17(d). Without performing any physical opera-

tions, new X-type stabilizer generators can be defined with the physical qubits prepared in the $|+\rangle$ state as their vertices. The newly defined stabilizer generators are shown in Fig. 17(e). Then, we perform X and Z stabilizer measurements, including the new stabilizer generators depicted in Fig. 17(f). After d rounds of this type of stabilizer measurements, the three patches fuse together as shown in Fig. 17(f). Finally, we measure the data qubits of the lower long portion of the fused patches to shorten its length. Eventually, three patches are fused into the densely packed configuration as shown in Fig. 17(g).

To separate patches from a densely packed state, one can achieve this by reversing the above procedure. In doing so, the operations to initialize the physical qubits to the $|0\rangle(|+\rangle)$ must be replaced with the operations to measure the physical qubits in the $Z(X)$ basis.

APPENDIX B: MOVEMENT METHOD FOR DENSELY PACKED SURFACE CODE

We present the deformation procedure for the movement of an entire row of the densely packed surface codes. Figure 18 illustrates the deformation procedure in a condensed manner. The red lines correspond to X boundaries and the blue lines correspond to Z boundaries. The physical operations required for the deformation, such as initialization of the state of physical qubits, stabilizer measurements, and measurement of physical qubits, are the same as those described in Appendix A.

Figure 18(a) shows the densely packed surface code before the movement. The physical qubits located below the bottom Z boundary of the densely packed surface code are initialized in the $|0\rangle$ state, and stabilizer measurements are performed, including the newly introduced stabilizer generators. After d rounds of stabilizer measurements, the bottom of the densely packed surface code is extended, as shown in Fig. 18(b). This operation corresponds to the extension of the logical Z operator associated with the lower code word in the densely packed surface code.

Next, we initialize the physical qubits to the $|+\rangle$ state along the X boundaries connecting the top and bottom regions of the densely packed surface code. We perform stabilizer measurements, including new stabilizer generators. After d rounds of stabilizer measurements, we obtain the expanded surface codes shown in Fig. 18(c). This operation corresponds to extending the logical X operators of the upper two code words downward.

We measure the physical qubits in the Z basis to shorten the logical Z operator of the lower code word, obtaining the state shown in Fig. 18(d). Similarly, we measure the physical qubits in the X basis to shorten the logical operators of the upper two code words. Figure 18(e) shows the result, and the entire densely packed surface code has moved downward from the state shown in Fig. 18(a). From the above, one can move the entire densely packed row downward with $2d$ rounds plus two parallel physical qubit measurement steps.

An alternative way to move the densely packed surface code is to repeatedly fuse and separate patches. In this approach, the lower patches are separated and shifted downward, followed by re-fusing the upper patches. However, this method requires an additional d rounds compared to the procedure in Fig. 18, and thus we adopt the latter.

[1] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM J. Comput. **26**, 1484 (1997).

[2] C. Gidney and M. Ekerå, How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits, Quantum **5**, 433 (2021).

[3] B. Bauer, S. Bravyi, M. Motta, and G. K.-L. Chan, Quantum algorithms for quantum chemistry and quantum materials science, Chem. Rev. **120**, 12685 (2020).

[4] D. P. DiVincenzo, The physical implementation of quantum computation, Fortschr. Phys. **48**, 771 (2000).

[5] P. W. Shor, Scheme for reducing decoherence in quantum computer memory, Phys. Rev. A **52**, R2493 (1995).

[6] J. Preskill, Reliable quantum computers, Proc. R. Soc. A **454**, 385 (1998).

[7] B. M. Terhal, Quantum error correction for quantum memories, Rev. Mod. Phys. **87**, 307 (2015).

[8] A. Y. Kitaev, Fault-tolerant quantum computation by anyons, Ann. Phys. (NY) **303**, 2 (2003).

[9] R. Acharya, L. Aghababaie-Beni, I. Aleiner, T. I. Andersen, M. Ansmann, F. Arute, K. Arya, A. Asfaw, N. Astrakhantsev, J. Atalaya, *et al.*, Quantum error correction below the surface code threshold, Nature (London) **638**, 920 (2025).

[10] X. Xue, M. Russ, N. Samkharadze, B. Undseth, A. Sammak, G. Scappucci, and L. M. Vandersypen, Quantum logic with spin qubits crossing the surface code threshold, Nature (London) **601**, 343 (2022).

[11] S. B. Bravyi and A. Y. Kitaev, Quantum codes on a lattice with boundary, arXiv:quant-ph/9811052.

[12] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Surface codes: Towards practical large-scale quantum computation, Phys. Rev. A **86**, 032324 (2012).

[13] D. Horsman, A. G. Fowler, S. Devitt, and R. Van Meter, Surface code quantum computing by lattice surgery, New J. Phys. **14**, 123011 (2012).

[14] A. G. Fowler and C. Gidney, Low overhead quantum computation using lattice surgery, arXiv:1808.06709.

[15] D. Litinski and F. von Oppen, Lattice surgery with a twist: Simplifying Clifford gates of surface codes, Quantum **2**, 62 (2018).

[16] D. Litinski, A game of surface codes: Large-scale quantum computing with lattice surgery, Quantum **3**, 128 (2019).

[17] H. Bombin and M. A. Martin-Delgado, Optimal resources for topological two-dimensional stabilizer codes: Comparative study, Phys. Rev. A **76**, 012305 (2007).

[18] J. T. Anderson, Homological stabilizer codes, Ann. Phys. (NY) **330**, 1 (2013).

[19] Y. Tomita and K. M. Svore, Low-distance surface codes under realistic quantum noise, Phys. Rev. A **90**, 062320 (2014).

[20] A. R. O'Rourke and S. Devitt, Compare the pair: Rotated versus unrotated surface codes at equal logical error rates, Phys. Rev. Res. **7**, 033074 (2025).
[21] M. B. Hastings and A. Geller, Reduced space-time and time costs using dislocation codes and arbitrary ancillas, Quantum Inf. Comput. **15**, 962 (2015).
[22] S. Nagayama, T. Satoh, and R. Van Meter, State injection, lattice surgery, and dense packing of the deformation-based surface code, Phys. Rev. A **95**, 012321 (2017).
[23] C. Gidney, M. Newman, P. Brooks, and C. Jones, Yoked surface codes, Nat. Commun. **16**, 4498 (2025).
[24] D. Copsey, M. Oskin, F. T. Chong, I. Chuang, and K. Abdel-Gaffar, Memory hierarchies for quantum data, in *1st Workshop on Non-Silicon Computing* (2002).
[25] M. Oskin, F. Chong, and I. Chuang, A practical architecture for reliable quantum computers, Computer **35**, 79 (2002).
[26] D. Thaker, T. Metodi, A. Cross, I. Chuang, and F. Chong, Quantum memory hierarchies efficient designs to match available parallelism in quantum computing, in *33rd International Symposium on Computer Architecture (ISCA '06)* (IEEE, New York, 2006), pp. 378–390.
[27] T. Kobori, Y. Suzuki, Y. Ueno, T. Tanimoto, S. Todo, and Y. Tokunaga, Lsqca: Resource-efficient load/store architecture for limited-scale fault-tolerant quantum computing, in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)* (IEEE, New York, 2025), pp. 304–320.
[28] D. Gottesman, Stabilizer codes and quantum error correction, arXiv:quant-ph/9705052.
[29] D. Wang, A. Fowler, A. Stephens, and L. Hollenberg, Threshold error rates for the toric and planar codes, Quantum Inf. Comput. **10**, 456 (2010).
[30] I. Byun, J. Kim, D. Min, I. Nagaoka, K. Fukumitsu, I. Ishikawa, T. Tanimoto, M. Tanaka, K. Inoue, and J. Kim, Xqsim: Modeling cross-technology control processors for 10+k qubit quantum computers, in *Proceedings of the 49th Annual International Symposium on Computer Architecture (ISCA '22)* (ACM, New York, 2022), pp. 366–382.
[31] C. Duckering, J. M. Baker, D. I. Schuster, and F. T. Chong, Virtualized logical qubits: A 2.5d architecture for error-corrected quantum computing, in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (IEEE, New York, 2020), pp. 173–185.
[32] Y. Ueno, M. Kondo, M. Tanaka, Y. Suzuki, and Y. Tabuchi, Qulatis: A quantum error correction methodology toward lattice surgery, in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* (IEEE, New York, 2022), pp. 274–287.
[33] C. Chamberland and E. T. Campbell, Circuit-level protocol and analysis for twist-based lattice surgery, Phys. Rev. Res. **4**, 023090 (2022).
[34] C. Gidney, Cleaner magic states with hook injection, arXiv:2302.12292.
[35] H. Bombin and M. A. Martin-Delgado, Quantum measurements and gates by code deformation, J. Phys. A **42**, 095302 (2009).
[36] M. Beverland, V. Kliuchnikov, and E. Schoute, Surface code compilation via edge-disjoint paths, PRX Quantum **3**, 020342 (2022).
[37] C. Chamberland and E. T. Campbell, Universal quantum computing with twist-free and temporally encoded lattice surgery, PRX Quantum **3**, 010331 (2022).
[38] M. E. Beverland, P. Murali, M. Troyer, K. M. Svore, T. Hoefler, V. Kliuchnikov, G. H. Low, M. Soeken, A. Sundaram, and A. Vaschillo, Assessing requirements to scale to practical quantum advantage, arXiv:2211.07629.
[39] J. Lee, D. W. Berry, C. Gidney, W. J. Huggins, J. R. McClean, N. Wiebe, and R. Babbush, Even more efficient quantum computations of chemistry through tensor hypercontraction, PRX Quantum **2**, 030305 (2021).
[40] Y. Ueno, T. Saito, T. Tanimoto, Y. Suzuki, Y. Tabuchi, S. Tamate, and H. Nakamura, High-performance and scalable fault-tolerant quantum computation with lattice surgery on a 2.5D architecture, arXiv:2411.17519.
[41] C. Gidney, Stim: A fast stabilizer circuit simulator, Quantum **5**, 497 (2021).
[42] O. Higgott and C. Gidney, Sparse blossom: Correcting a million errors per core second with minimum-weight matching, Quantum **9**, 1600 (2025).
[43] K. Fujiu, Code for: Dense packing of the surface code, https://github.com/kohei-fujiu/Dense_Pakcing.
[44] Y. Hirano and K. Fujii, Locality-aware Pauli-based computation for local magic state preparation, in *2025 IEEE International Conference on Quantum Computing and Engineering (QCE)* (IEEE, New York, 2025), Vol. 1, pp. 670–680.