

Subsystem many-hypercube codes: High-rate concatenated codes with low-weight syndrome measurements

Ryota Nakai^{1,*} and Hayato Goto^{1,2}

¹*RIKEN Center for Quantum Computing (RQC), Wako, Saitama 351-0198, Japan*

²*Corporate Laboratory, Toshiba Corporation, Kawasaki, Kanagawa 212-8582, Japan*



(Received 24 October 2025; revised 18 November 2025; accepted 10 December 2025; published 13 January 2026)

Quantum error-correcting codes (QECCs) require high encoding rates in addition to high thresholds unless a sufficiently large number of physical qubits are available. Many-hypercube (MHC) codes, defined as concatenations of the $[[6, 4, 2]]$ quantum error-detecting code, have been proposed as high-performance and high-encoding-rate QECCs. However, concatenated codes have the disadvantage that the syndrome weight grows exponentially with respect to the concatenation level. To address this issue, here we propose subsystem quantum codes based on MHC codes. In particular, we study the smallest subsystem MHC codes, namely, subsystem codes derived from concatenated $[[4, 2, 2]]$ error-detecting codes. The resulting codes have a constant syndrome-measurement weight of 4, while keeping high encoding rates. We build block-MAP and neural-network decoders and show that they demonstrate superior performance to the bounded-distance decoder.

DOI: [10.1103/xbzn-vn37](https://doi.org/10.1103/xbzn-vn37)

I. INTRODUCTION

Quantum error-correcting codes (QECCs) [1,2] are widely accepted as a means of performing quantum computations fault-tolerantly. A quantum circuit on physical qubits is encoded into one on logical qubits, protected from noise by the repetition of detecting and correcting errors via syndrome measurements. The surface code is the current standard QECC [3–5]. The advantage of the surface code lies in the fact that it has a high error threshold and is implemented locally via nearest-neighbor interaction on a two-dimensional array of qubits. However, fault-tolerant quantum computations with it require high space overheads, that is, one needs d^2 physical qubits to encode a single logical qubit with code distance d . Bivariate bicycle (BB) codes, which are representative of the quantum low-density parity-check (qLDPC) codes, have been proposed as a solution to this problem [6]. With the same code distance, the encoding rate is much higher than that of the surface code. The syndrome measurements in the BB code involve weight-6 check operators, but, unlike the surface code, they are nonlocal: one needs to interact two qubits far apart.

Another solution to the encoding-rate problem has been proposed based on code concatenation [7–9]. Here, we refer to the number of codes that are concatenated to construct a single code block as concatenation level or simply level, which is denoted by r in the following. The concatenation of the quantum Hamming codes [10,11] of sizes $7, 15, 31, \dots, 2^{r+2} - 1$ leads to an asymptotically constant encoding rate of $1/36$ as the concatenation level increases [7]. The concatenation of the $[[6, 4, 2]]$ quantum error-detecting code, the resultant codes of which are referred to as the $[[6, 4, 2]]$ many-hypercube (MHC) codes [8,12,13], has also been proposed. Within intermediate sizes, $[[6, 4, 2]]$ MHC codes have a high encoding rate of $(4/6)^r$. In such concatenated codes, however, the weights of the syndromes (check operators) grow exponentially as the concatenation level increases. This is because the weights of lower-level logical operators, which make up the check operators, also grow exponentially. Specifically, the level- r concatenated quantum Hamming code has check operators of weight as high as $4 \cdot 6^{r-1}$ [7], and the level- r $[[6, 4, 2]]$ MHC code has check operators of weight $6 \cdot 2^{r-1}$ [8].

A similar problem occurred in the Shor code [14], which is a concatenated code of the three-qubit repetition codes for bit-flip and phase-flip errors. When the former code is defined using logical qubits of the latter code, the syndrome measurements for bit-flip errors are performed on six qubits, while those for phase-flip errors on two qubits. The weight of the check operators can be reduced by making part of the qubits not engaged in quantum

*Contact author: rnakai@riken.jp

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

computations. The resulting code is known as the Bacon-Shor code, which is a type of subsystem code [15–17]. The redundant parts of the qubits are referred to as the gauge qubits. Since any two operators are equivalent modulo gauge-qubit operators, the weight of the check operators is reduced to 2 in the Bacon-Shor code. Another method of reducing the check-operator weight is by introducing time-periodic syndrome measurements, known as Floquet codes [18–26].

In this work, we propose subsystem codes made from MHC codes [8], which are referred to as subsystem MHC codes. As a simple example, we study the $[[4, 2, 2]]$ subsystem MHC codes, which are the subsystem codes of the concatenated $[[4, 2, 2]]$ error-detecting codes. The $[[4, 2, 2]]$ code is the smallest stabilizer code that can detect a single-qubit error [27,28]. The resulting code at concatenation level r is a $[[4^r, 2^r, g, 2^r]]$ code, where $g = 4^r + 2^r - 2 \cdot 3^r$ is the number of gauge qubits. The syndrome measurements are performed on four qubits at an arbitrary level, while the original level- r $[[4, 2, 2]]$ MHC code has check operators of weight as high as $4 \cdot 2^{r-1}$. However, the decoding becomes problematic due to the large number of gauge qubits. We overcome this problem by building two kinds of decoders for the new codes: a block-MAP (maximum-a-posteriori) decoder [29] up to level 3 and a neural-network (NN) decoder [30,31] up to level 4. We evaluate the performance of the $[[4, 2, 2]]$ subsystem MHC codes and show that they outperform the theoretical performance of the bounded-distance (BD) decoder and also the performance evaluated with belief propagation with ordered statistics decoding (BP-OSD) [32].

In the following, we denote the $[[4, 2, 2]]$ and $[[6, 4, 2]]$ error-detecting codes by D_4 and D_6 , respectively, and their concatenated codes by $D_{n_1, n_2, \dots, n_r}$ when the level- i code is D_{n_i} , following the notation of [12]. Specifically, the original level- r $[[4, 2, 2]]$ MHC codes are denoted by D_{4^r} (e.g., $D_{4,4}$ for level 2 and $D_{4,4,4}$ for level 3) and we refer to the level- r subsystem ones as the subsystem D_{4^r} codes.

The rest of this paper is organized as follows. In Sec. II, we explain the definition of the subsystem D_{4^r} codes in more detail. In Sec. III, we present our decoders and show their performance. Finally, we present our conclusions in Sec. IV.

II. $[[4, 2, 2]]$ SUBSYSTEM MHC CODES

A. D_4 code

We start with the definition of the D_4 code [27,28]. The D_4 code is a stabilizer code defined on four qubits and has two stabilizer generators given by

$$SX^{(1)} = X_1 X_2 X_3 X_4, \quad SZ^{(1)} = Z_1 Z_2 Z_3 Z_4, \quad (1)$$

where X_j and Z_j are the Pauli operators on qubit j , and the number in the superscript indicates the level of

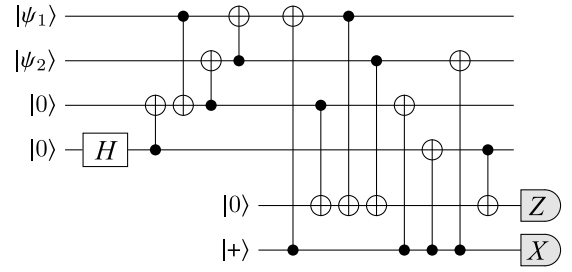


FIG. 1. A circuit for an encoder of $|\psi_1 \psi_2\rangle_L$ state and the syndrome measurement of the D_4 code.

concatenation. The logical operators of the two encoded qubits are given by

$$\begin{cases} X_{L1}^{(1)} = X_2 X_3 \\ Z_{L1}^{(1)} = Z_1 Z_2, \end{cases} \quad \begin{cases} X_{L2}^{(1)} = X_1 X_2 \\ Z_{L2}^{(1)} = Z_2 Z_3. \end{cases} \quad (2)$$

The code space is spanned by

$$|\psi_1 \psi_2\rangle_L = \left(X_{L1}^{(1)}\right)^{\psi_1} \left(X_{L2}^{(1)}\right)^{\psi_2} |00\rangle_L, \quad (3)$$

where $\psi_1, \psi_2 = 0, 1$ and $|00\rangle_L = (|0000\rangle + |1111\rangle)/\sqrt{2}$ is the Greenberger-Horne-Zeilinger state. A quantum circuit for an encoder [33] followed by the syndrome measurement of (1) using two ancilla qubits [34] for the D_4 code is shown in Fig. 1. This syndrome extraction method can detect harmful errors that spread into data qubits [34].

B. $D_{4,4}$ code

Let the four physical qubits of the D_4 code be aligned along the x axis from $x = 1$ to 4. At level 2, we prepare four sets of the D_4 code, each of which is specified by its position along the y axis from $y = 1$ to 4. The $D_{4,4}$ code is then defined on $4^2 = 16$ physical qubits on two-dimensional lattice sites [8].

The logical operators and part of the stabilizer generators of $D_{4,4}$ are constructed using the logical operators of D_4 . There are two logical qubits per each D_4 code, for a total of eight logical qubits. The $D_{4,4}$ code consists of two D_4 codes, each of which is defined by four level-1 logical qubits. Let the j th level-1 logical operators at y be denoted by $X_{Lj,y}^{(1)}$ and $Z_{Lj,y}^{(1)}$, where $j \in [1, 2]$ and $y \in [1, 4]$. One obtains the level-2 stabilizer generators [Fig. 2(a)]

$$SX_j^{(2)y} = \bigotimes_{y=1}^4 X_{L3-j,y}^{(1)}, \quad SZ_j^{(2)y} = \bigotimes_{y=1}^4 Z_{Lj,y}^{(1)}, \quad (4)$$

and the logical operators

$$X_{Ljk}^{(2)} = \bigotimes_{y=3-k}^{4-k} X_{Lj,y}^{(1)}, \quad Z_{Ljk}^{(2)} = \bigotimes_{y=k}^{k+1} Z_{Lj,y}^{(1)}, \quad (5)$$

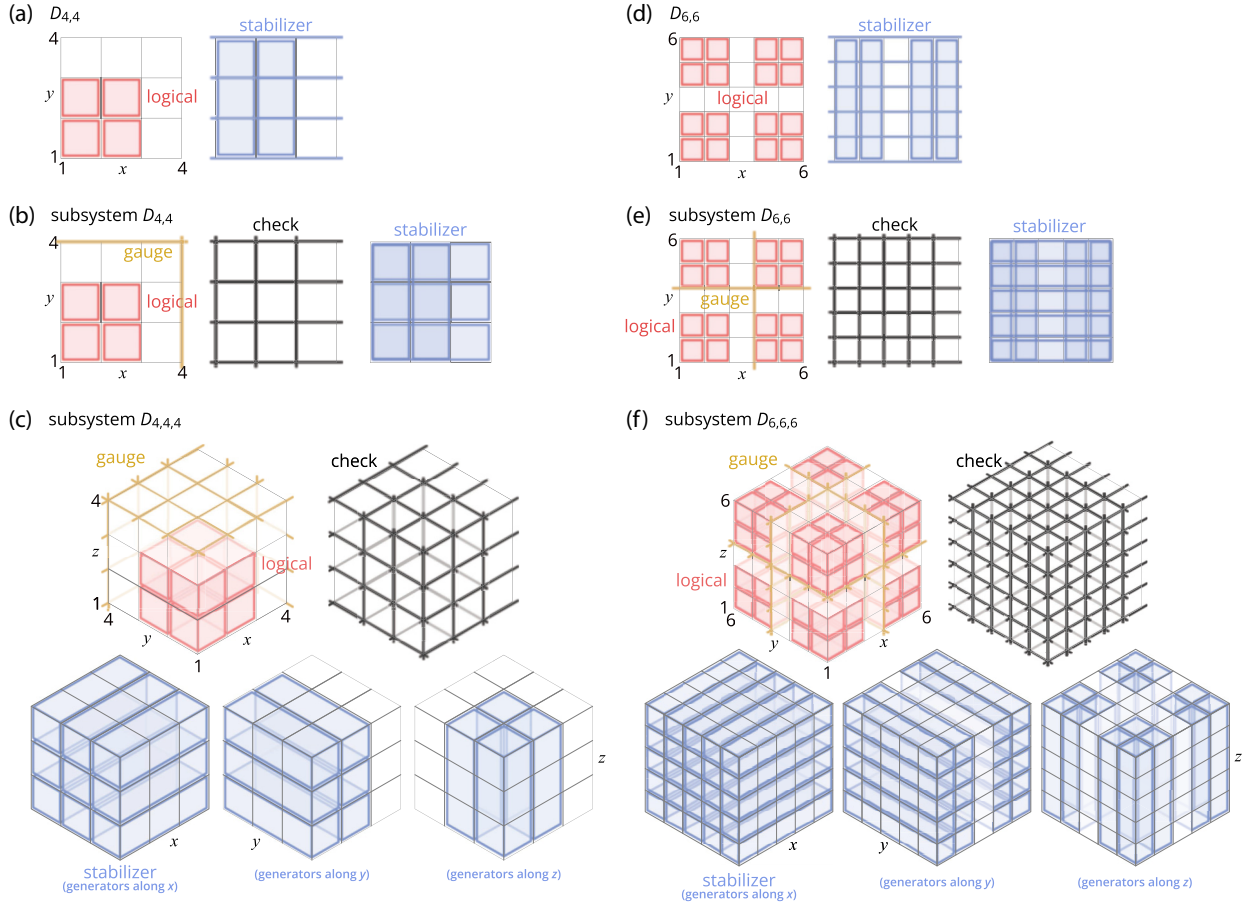


FIG. 2. The logical-qubit operators, gauge-qubit operators, stabilizer generators, and check operators of (a) $D_{4,4}$, (b) subsystem $D_{4,4}$, (c) subsystem $D_{4,4,4}$, (d) $D_{6,6}$, (e) subsystem $D_{6,6}$, and (f) subsystem $D_{6,6,6}$ codes are shown. In (c),(f), the stabilizer generators that extend along the x , y , and z directions are shown separately. The logical-qubit (gauge-qubit) operators are defined on qubits shown by red squares or cubes (yellow lines). Notice that the logical X and Z operators of the same logical or gauge qubits are not defined on the same sets of qubits; they share an odd number of qubits so that they anticommute with each other. Two types of stabilizer generators, SX and SZ , are defined on the same sets of qubits shown by blue lines, rectangles or cuboids. This is also the case for the X and Z check operators (black lines).

where $j, k = 1, 2$. Here y in the superscripts in (4) indicates that the stabilizer generators extend along the y axis [Fig. 2(a)]. The stabilizer generators with the same subscript are made to be defined on the same set of physical qubits. Combined with the level-1 stabilizer generators, there are 12 stabilizer generators and four logical qubits that have weight-4 logical operators, forming a $[[4^2, 2^2, 2^2]]$ code. Eight of the 12 stabilizer generators are weight-4 operators and the remaining ones given by (4) are weight-8 operators [Fig. 2(a)].

C. Subsystem $D_{4,4}$ code

An $[[n, k, g, d]]$ subsystem code encodes k -qubit states on n physical qubits with code distance d . It contains g gauge qubits. We construct a $[[4^2, 2^2, 2, 2^2]]$ subsystem code from the $D_{4,4}$ code by assigning four-qubit operators at $x = 4$ and $y = 4$ as the gauge-qubit operators [Fig. 2(b)]. Specifically, they are given by the level-1 stabilizer

generators $SX_4^{(1)}, SZ_4^{(1)}$ at $y = 4$ and two operators

$$\bigotimes_{y=1}^4 X_{4y}, \quad \bigotimes_{y=1}^4 Z_{4y} \quad (6)$$

at $x = 4$, where X_{xy} and Z_{xy} are Pauli operators of the physical qubit at (x, y) . The gauge-qubit operators do not share physical qubits with the level-2 logical-qubit operators (5). Since these operators are now irrelevant to the code, the level-2 stabilizer generators (4) are equivalent to weight-4 check operators [Fig. 2(b)]. As a result, there are 14 independent check operators with weight 4 in the subsystem $D_{4,4}$ code.

The check operators form a gauge group $\mathcal{G}$, from which a subsystem code is defined [16,26]. The stabilizer group is the Abelian subgroup of $\mathcal{G}$ defined by $\mathcal{Z}(\mathcal{G}) \cap \mathcal{G}$, where $\mathcal{Z}(\mathcal{G})$ is the centralizer of $\mathcal{G}$ [2]. The stabilizer generators

are weight-8 operators, which are the products of two adjacent parallel check operators [Fig. 2(b)]. Therefore, we can determine the syndrome outcome by measuring the weight-4 check operators and calculating the products of the measurement outcome. Notice that while the measurement outcome of the check operators is determined probabilistically since they are not commutable with each other, their products are not. There are 10 stabilizer generators in the subsystem $D_{4,4}$ code. The logical-qubit operators are part of the generators of $\mathcal{Z}(\mathcal{G})$ that are not included in $\mathcal{G}$, and are the same as those of the $D_{4,4}$ code (5).

D. Subsystem D_{4^r} codes

The subsystem D_{4^r} code is defined on 4^r qubits forming an r -dimensional hyperlattice of linear dimension 3. Along each axis, there are 4^{r-1} parallel lines that connect four qubits, for a total of $r \cdot 4^{r-1}$ lines. We define weight-4 check operators as the products of X and Z on these lines, respectively. The gauge group $\mathcal{G}$ is generated by the check operators. The stabilizer generators of $\mathcal{S} = \mathcal{Z}(\mathcal{G}) \cap \mathcal{G}$ are operators on hypercuboids of $4 \cdot 2^{r-1}$ physical qubits (four in one of the r directions and two in the others). For each direction, there are $2 \cdot 3^{r-1}$ stabilizer generators. However, not all the stabilizer generators defined in this way are independent; there are $2(3^r - 2^r)$ independent stabilizer generators in the subsystem D_{4^r} code. The syndrome outcome of the stabilizer generators (weight 2^{r+1}) is determined by the products of the measurement outcome of 2^{r-1} check operators (weight 4). This indicates that the measurement weight is reduced from 2^{r+1} to 4 by making the D_{4^r} code into a subsystem code.

The remaining generators of $\mathcal{Z}(\mathcal{G})$ not included in $\mathcal{S}$ form 2^r logical qubits, whose operator weight is 2^r . The remaining generators of $\mathcal{G}$ not included in $\mathcal{S}$ form gauge qubits. In total, there are 2^r logical qubits, $2(3^r - 2^r)$ stabilizer generators, and $4^r + 2^r - 2 \cdot 3^r$ gauge qubits.

The operators of the subsystem D_{4^r} code are defined recursively from the lower-level codes. Let the stabilizer generators and logical-qubit operators of the subsystem $D_{4^{r-1}}$ code at $x_r \in [1, 4]$ be denoted by

$$SA_{j_1 \dots \check{j}_n \dots j_{r-1}, x_r}^{(r-1)x_n}, \quad A_{Lj_1 \dots \check{j}_n \dots j_{r-1}, x_r}^{(r-1)} \quad (7)$$

where $n \in [1, r-1]$ is the direction along which the operator extends and $A = X, Z$. Here, $j_1 \dots \check{j}_n \dots j_r$ indicates a set of indices obtained from $j_1 \dots j_r$ by omitting j_n , where $j_1, \dots, j_{n-1}$ take values 1 and 2 while $j_{n+1}, \dots, j_r$ run from 1 to 3. On the other hand, the subscript indices of the logical operators take values 1 and 2.

From these operators, the stabilizer generators at level r are given by

$$SA_{j_1 \dots \check{j}_n \dots j_r}^{(r)x_n} = \bigotimes_{x_r=j_r}^{j_r+1} SA_{j_1 \dots \check{j}_n \dots j_{r-1}, x_r}^{(r-1)x_n}, \quad (8)$$

for $n \in [1, r-1]$, $j_r \in [1, 3]$, and

$$SX_{j_1 \dots \check{j}_{r-1}, j_r}^{(r)x_r} = \bigotimes_{x_r=1}^4 X_{L3-j_1 \dots 3-j_{r-1}, x_r}^{(r-1)}, \quad (9)$$

$$SZ_{j_1 \dots \check{j}_{r-1}, j_r}^{(r)x_r} = \bigotimes_{x_r=1}^4 Z_{Lj_1 \dots j_{r-1}, x_r}^{(r-1)}. \quad (10)$$

The logical-qubit operators at level r are given by

$$X_{Lj_1 \dots j_r}^{(r)} = \bigotimes_{x_r=3-j_r}^{4-j_r} X_{Lj_1 \dots j_{r-1}, x_r}^{(r-1)}, \quad (11)$$

$$Z_{Lj_1 \dots j_r}^{(r)} = \bigotimes_{x_r=j_r}^{j_r+1} Z_{Lj_1 \dots j_{r-1}, x_r}^{(r-1)}, \quad (12)$$

for $j_r \in [1, 2]$. We choose the gauge-qubit operators within the hyperplane at $x_n = 4$ ($n \in [1, r]$) so that there is no intersection between the gauge- and logical-qubit operators.

Specifically, the operators of the subsystem $D_{4,4,4}$ code are shown in Fig. 2(c). The code is defined on $4^3 = 64$ physical qubits. There are 8 logical qubits, the operators of which are defined on $2 \times 2 \times 2$ physical qubits, 38 stabilizer generators consisting of 18 operators that extend along the x direction and are defined on $4 \times 2 \times 2$ physical qubits, 12 along the y direction on $2 \times 4 \times 2$ physical qubits, and 8 along the z direction on $2 \times 2 \times 4$ physical qubits. In addition, there are 18 gauge qubits.

E. Encoding and syndrome measurements of D_{4^r}

For encoding, one needs to fix the gauge to specify the encoded states. Specifically, we encode a physical state within the codespace of the original D_{4^r} codes following [8]. Due to the absence of the gauge qubits in the original codes, there is a one-to-one correspondence between the physical and encoded states, that is, we can fix the gauge degrees of freedom.

We measure check operators parallel to each direction in order from x_1 to x_r . That is, we first measure the $2 \cdot 4^{r-1}$ check operators parallel to x_1 , then measure those parallel to x_2 , and so on. After measuring all the directions, a single step of the syndrome measurement is completed. For each line in the hyperlattice, we attach two ancilla qubits for the syndrome measurements of X and Z check operators, respectively, as in Fig. 1. We need $2 \cdot 4^{r-1}$ ancilla qubits to measure all the check operators parallel to each other. Since we can reuse the ancilla qubits for measuring check operators in the other directions, the space overhead due to the syndrome measurements is $3/2$.

The syndrome outcome is the eigenvalues of the stabilizer generators and agrees with the products of the

measurement outcome of the check operators. Hence, we need to measure only the weight-4 check operators in the syndrome measurements.

F. Subsystem D_{6^r} codes

The construction of the subsystem D_{6^r} codes from the original D_{6^r} codes is done in a similar way to that of the subsystem D_{4^r} codes. Compared with the original D_{6^r} codes, the syndrome-measurement weight is reduced from $6 \cdot 2^{r-1}$ to 6.

The D_{6^r} codes are defined on qubits placed on an r -dimensional hyperlattice of linear dimension 5. There are $r \cdot 6^{r-1}$ lines of length 5. We define weight-6 check operators as the product of X and Z on these lines, which form the gauge group. The procedure of constructing the subsystem D_{6^r} codes from this point onward is similar to that of the subsystem D_{4^r} codes in Sec. II D. The resulting code has 4^r logical qubits, $2(5^r - 4^r)$ stabilizer generators, and $6^r + 4^r - 2 \cdot 5^r$ gauge qubits. Specifically, the operators of the $D_{6,6}$, subsystem $D_{6,6}$, and subsystem $D_{6,6,6}$ codes are shown schematically in Figs. 2(d)–2(f). We assign gauge-qubit operators within the hyperplanes at $x_n = 4$ ($n \in [1, r]$).

III. DECODERS

In the following, we focus on the subsystem D_{4^r} codes and evaluate their performance up to $r = 4$ assuming bitflip errors. The other errors such as measurement errors or circuit-level noise are not considered and left for future work. We used `stim` [35] to simulate the syndrome outcome.

Our codes have a large number of gauge qubits, the number of which grows exponentially with respect to the concatenation level. Compared with the original D_{4^r} codes, the number of stabilizer generators decreases by that of the gauge qubits, that is, less information is available for decoding. In addition, the amount of error degeneracy is $2^{\# \text{ of the gauge qubits}}$ times as large as that for the D_{4^r} codes since any two errors that are equivalent modulo gauge-qubit operators are regarded as the same error. This makes optimal or nearly optimal decoders, which enumerate all the error degeneracy, infeasible at least within reasonable calculation time. For this reason, we build an NN decoder to empirically learn the decoding up to level 4. In addition, we use the block-MAP decoder, which is optimal, up to level 3.

For comparison, we evaluate the logical error rate using BP-OSD [32], which is the standard for the qLDPC codes. We used the `bp-osd` package [36,37] with 200 iterations and OSD order 2. All the results are compared with the theoretical performance of the BD decoder that successfully corrects up to $d/2 - 1$ errors, where $d = 2^r$ is the code distance of the subsystem D_{4^r} code. The logical error rate of

the BD decoder is given by

$$P^{(r)}(p) = 1 - \sum_{j=0}^{2^{r-1}-1} \binom{4^r}{j} p^j (1-p)^{4^r-j}, \quad (13)$$

where p is the bitflip probability of each qubit. When $p \ll 1$, $P^{(r)}(p) \propto p^{2^{r-1}}$. BP-OSD shows better performance than the BD decoder at high bitflip probability, while the logical error rate does not obey $p^{2^{r-1}}$ at lower p [Figs. 3(a) and 3(b)]. The performance of BP-OSD deteriorates as the concatenation level increases [Fig. 3(b)].

A. Block-MAP decoder

Let us review the block-MAP decoder, which deduces the most probable error $E \in \mathcal{P}_n$ by evaluating the maximum a posteriori (MAP) probability for a given syndrome outcome $\mathbf{s}$ [29], where $\mathcal{P}_n$ is the n -qubit Pauli group. In evaluating the MAP probability, all the error operators that are equivalent modulo the gauge group need to be identified. In general, a bitflip error operator E can be decomposed as

$$E = L(E)T(E)G(E), \quad (14)$$

where $L(E)$ is a logical operator, $T(E)$ is a pure error operator, and $G(E)$ is an element of the gauge group $\mathcal{G}$. Since Pauli operators X and Z are completely decoupled in our codes, all the components $L(E)$, $T(E)$, and $G(E)$ are composed solely of X operators. The pure error $T(E)$ has a one-to-one correspondence with the syndrome outcome $\mathbf{s}$. Specifically, $T(E)$ is a product of pure error operators $\{TX_{j_1 \dots j_n}^{(r)x_n}\}$, each of which flips the eigenvalue of the stabilizer generator with the same indices following $\{SZ_{j_1 \dots j_n}^{(r)x_n}, TX_{j_1 \dots j_n}^{(r)x_n}\} = 0$, but has no other effect. Here, $\{SZ_{j_1 \dots j_n}^{(r)x_n}\}$ are the level- r stabilizer generators in Eqs. (8) and (10), the eigenvalues of which are the syndrome outcome $\mathbf{s}^{(r)}$. The logical operator $L(E)$ is specified by a set of indices $\mathbf{l}^{(r)}$ by

$$L(E) = \prod_{j_1 \dots j_r} \left(X_{Lj_1 \dots j_r}^{(r)} \right)^{l_{j_1 \dots j_r}^{(r)}}. \quad (15)$$

As a result, the problem of finding the most probable error operator E reduces to finding the maximum conditional probability $P(\mathbf{l}^{(r)}|\mathbf{s}^{(r)})$ of $\mathbf{l}^{(r)}$ from a given syndrome outcome $\mathbf{s}^{(r)}$. Since $P(\mathbf{l}^{(r)}|\mathbf{s}^{(r)}) = P(\mathbf{l}^{(r)}, \mathbf{s}^{(r)}) / \sum_{\mathbf{l}^{(r)}} P(\mathbf{l}^{(r)}, \mathbf{s}^{(r)})$, the maximum conditional probability is equivalent to the maximum joint probability. As the probability is evaluated exhaustively, the performance of the block-MAP decoder must be optimal [29].

Here, we decompose the syndrome outcome $\mathbf{s}^{(r)}$ into $\mathbf{s}^{(r)x_1 \dots x_{r-1}}$ and $\mathbf{s}^{(r)x_r}$, which correspond to Eqs. (8) and (10),

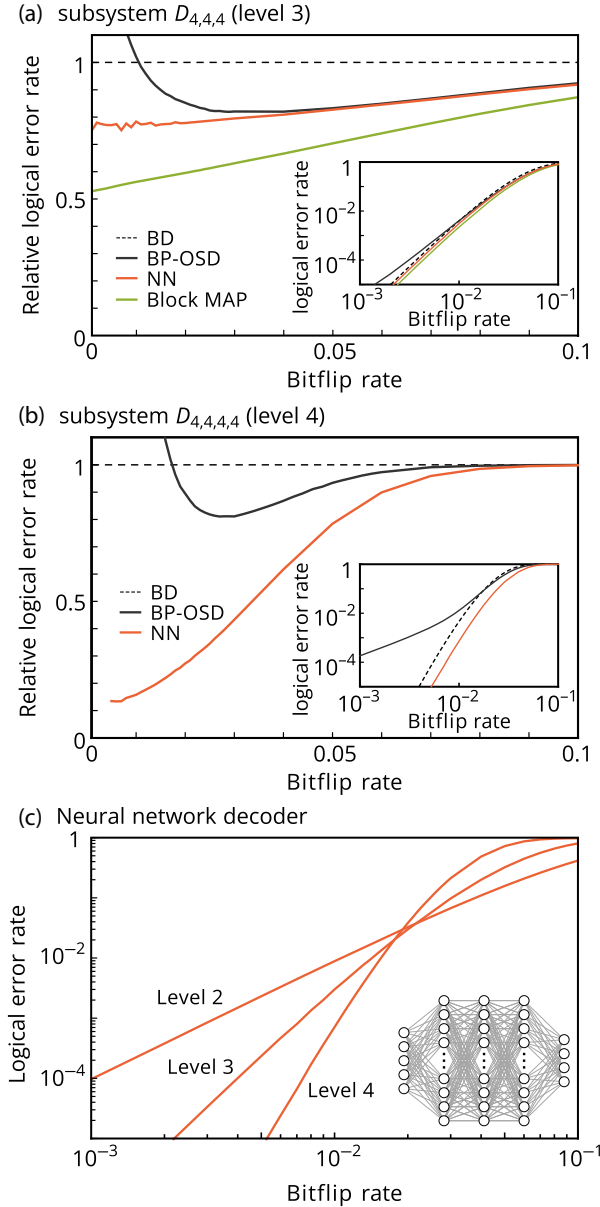


FIG. 3. The performance of (a) the subsystem $D_{4,4,4}$ code, (b) the subsystem $D_{4,4,4,4}$ code, and (c) the subsystem $D_{4,4}$ to $D_{4,4,4,4}$ codes with the neural network (NN) decoder. In (a),(b), the dashed lines are the theoretical performance of the bounded-distance (BD) decoder (13) and the solid lines are numerically evaluated by belief propagation with ordered statistics decoding (BP-OSD) (black), the NN decoder (red), and the block-MAP decoder (green), respectively. The vertical axis of (a),(b) is the relative logical error rate compared with that of the BD decoder (13), while the insets show the bare logical error rate on a log scale. In (c), we plot the logical error rate estimated with the NN decoder from level 2 to 4. The inset shows the schematic picture of the NN at level 2.

respectively. As we saw in Sec. IID, the level- r operators are constructed from the level- $(r-1)$ ones. Let the indices of the level- $(r-1)$ codes be denoted by $\mathbf{I}^{(r-1)}$ and $\mathbf{s}^{(r-1)}$.

From Eqs. (10) and (11), $\mathbf{s}^{(r)x_r}$ and $\mathbf{I}^{(r)}$ are determined by $\mathbf{I}^{(r-1)}$, while $\mathbf{s}^{(r)x_1 \dots x_{r-1}}$ is determined by $\mathbf{s}^{(r-1)}$. This reduces to a recursive relation of the joint probability given by [29]

$$P(\mathbf{I}^{(r)}, \mathbf{s}^{(r)}) = \sum_{\mathbf{I}^{(r-1)}} \delta(\mathbf{I}^{(r)}, \mathbf{I}^{(r-1)}) \delta(\mathbf{s}^{(r)x_r}, \mathbf{I}^{(r-1)}) \times \sum_{\mathbf{s}^{(r-1)}} \delta(\mathbf{s}^{(r)x_1 \dots x_{r-1}}, \mathbf{s}^{(r-1)}) P(\mathbf{I}^{(r-1)}, \mathbf{s}^{(r-1)}), \quad (16)$$

where δ denotes the indicator function, which is 1 when two sets of indices are consistent and 0 otherwise.

We numerically evaluate the joint probability and decode the subsystem D_{4^r} codes up to $r = 3$. The logical error rate is proportional to $p^{2^{r-1}}$ and is smaller than the other decoders as it should be [Fig. 3(a)]. Decoding the subsystem $D_{4,4,4,4}$ code with the block-MAP decoder was not available within a reasonable calculation time.

B. Neural-network decoder

To learn the decoding of the subsystem D_{4^r} codes, we use fully connected deep neural networks [30,38–41]. The NNs learn the relation between the syndrome outcome and the indices of the most probable set of errors. We assign the syndrome outcome $\mathbf{s}^{(r)}$ as the input and the indices $\mathbf{I}^{(r)}$ for the logical operator $L(E)$ in Eq. (15) as the output. This class of decoders are called as high-level decoders in the sense that the output is not errors of the individual physical qubits but those of the logical qubits [38]. We train the NNs to minimize the binary cross entropy between the predicted $\mathbf{I}^{(r)}$, which takes real values between 0 and 1, and their ground truth, a set of 0s or 1s. As we assume bitflip errors, there are $3^r - 2^r$ input nodes and 2^r output nodes at level r .

The NNs are trained using sets of the syndrome outcome and logical indices simulated by a circuit with a specific bitflip probability p . The same NNs are used for decoding circuits with general p . Specifically, we consider a fully connected NN with three hidden layers with 16×3^r nodes. The numbers of hidden layers and nodes are determined so that the performance is well converged at level $r = 2, 3$ and is the best at level $r = 4$ [42]. We trained the NNs with 2^{22} samples simulated with $p = 0.04$ at level 2, 2^{23} samples with $p = 0.04$ at level 3, and 2^{29} samples with $p = 0.03$ at level 4.

The logical error rate of the NN decoder is better than that of the BD decoder, which indicates that the NN successfully corrects more errors than the bounded distance. In addition, the NN decoder performs better than BP-OSD across the entire range of p . However, considering the performance difference between the block-MAP and NN decoders at level 3 [Fig. 3(a)], the NN decoder may not fully utilize the potential of the subsystem D_{4^r} codes, which we leave for future work. The threshold for the bitflip error is estimated about 2%.

IV. CONCLUSIONS

In this work, we have proposed subsystem MHC codes as subsystem codes of MHC codes. In the case of the $[[4, 2, 2]]$ subsystem MHC codes, the weight of the check operators is reduced to 4 regardless of the concatenation level, and hence the problem of an exponentially large operator weight common to concatenated codes is resolved. In addition, the net encoding rate including ancilla qubits at level r is $1/(3 \cdot 2^{r-1})$, which inherits a high encoding rate from the original MHC codes. The subsystem MHC codes proposed here can be extended to arbitrary MHC codes including those with the $[[6, 4, 2]]$ code [8].

Due to the presence of an exponentially growing number of gauge qubits, the numerical cost of exhaustive searches, such as brute-force or block-MAP decoding, grows far faster than the original $[[4, 2, 2]]$ MHC codes as the concatenation level increases. Thus, we have built a fully connected deep NN decoder to reduce the numerical cost in decoding. The performance of the NN decoder exceeds that of the BD decoder and BP-OSD. Compared with the original $[[4, 2, 2]]$ MHC codes, the estimated threshold for the bitflip error is reduced from 7% to 2% (see Appendix). This reduction could be due to reduced information from the syndrome measurements, and also due to the decoder that is not sufficiently optimized. Indeed, the performance of the NN decoder has not reached that of the block-MAP (optimal) decoder. Closing this gap could potentially be the focus of future work.

Compared with the teleportation-based error correction proposed in Ref. [8], the advantages of the proposed syndrome extraction by ancilla measurements are that the number of ancillas is reduced by a factor of $1/4$ ($1/6$) for subsystem D_{4^r} (D_{6^r}) codes, and that this method is free from the need to prepare logical Bell states. However, syndrome extraction with ancillas in general requires repeated measurements, which would make the time overhead larger than that for teleportation-based error correction. In addition, entangling data qubits with ancillas could lead to error spreading, which is not considered in this work. We will discuss the circuit-level noise separately in a forthcoming paper.

ACKNOWLEDGMENTS

This work was supported by JST Moonshot R&D Grant No. JPMJMS2061.

DATA AVAILABILITY

The data that support the findings of this article are not publicly available upon publication because it is not technically feasible and/or the cost of preparing, depositing, and hosting the data would be prohibitive within the terms of this research project. The data are available from the authors upon reasonable request.

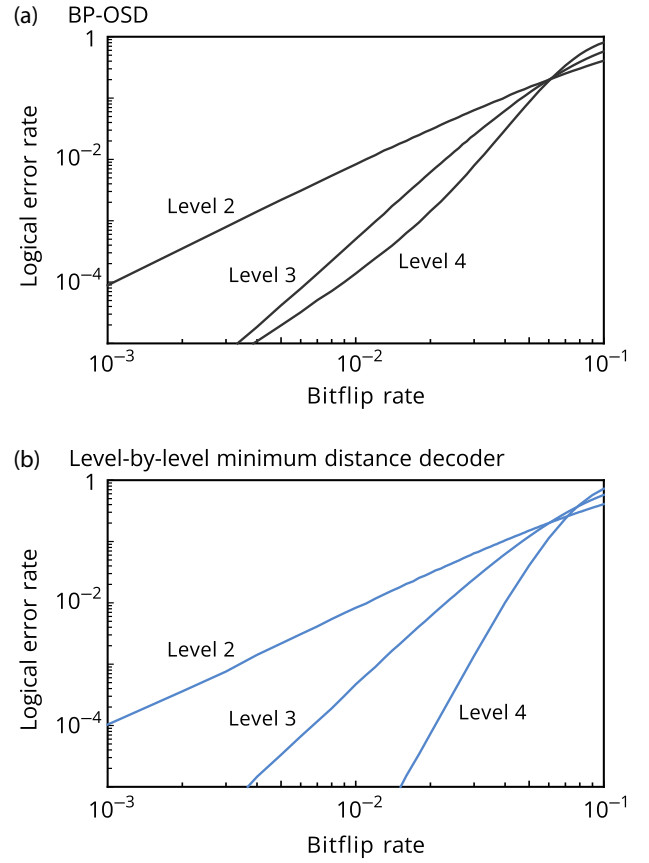


FIG. 4. The performance of the D_{4^r} codes by (a) BP-OSD and (b) the level-by-level minimum distance decoder [8]. Numerically evaluated logical error rates are plotted from level 2 to level 4.

APPENDIX: D_{4^r} CODES

In this appendix, we show the performance of the original D_{4^r} codes.

The original and subsystem D_{4^r} codes share the same logical-qubit operators, while the stabilizer generators are slightly different. Unlike the subsystem D_{4^r} codes, the original one inherits the lower-level stabilizer generators after concatenation. In total, there are $4^r - 2^r$ stabilizer generators, which constitute a $[[4^r, 2^r, 2^r]]$ stabilizer code.

We numerically evaluate the performance of the D_{4^r} codes, assuming bitflip error, in Fig. 4. We employ the BP-OSD and the level-by-level minimum distance (MD) decoder [8]. The threshold for the bitflip error estimated with the MD decoder is about 7%.

-
- [1] D. Gottesman, Surviving as a quantum computer in a classical world (2024). <https://www.cs.umd.edu/class/spring2024/cmssc858G/>.
 - [2] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, New York, 2010).

- [3] A. Kitaev, Fault-tolerant quantum computation by anyons, *Ann. Phys.* **303**, 2 (2003).
- [4] S. B. Bravyi and A. Y. Kitaev, Quantum codes on a lattice with boundary, [arXiv:quant-ph/9811052 \[quant-ph\]](https://arxiv.org/abs/quant-ph/9811052).
- [5] 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).
- [6] S. Bravyi, A. W. Cross, J. M. Gambetta, D. Maslov, P. Rall, and T. J. Yoder, High-threshold and low-overhead fault-tolerant quantum memory, *Nature* **627**, 778 (2024).
- [7] H. Yamasaki and M. Koashi, Time-efficient constant-space-overhead fault-tolerant quantum computation, *Nat. Phys.* **20**, 247 (2024).
- [8] H. Goto, High-performance fault-tolerant quantum computing with many-hypercube codes, *Sci. Adv.* **10**, eadp6388 (2024).
- [9] S. Yoshida, S. Tamiya, and H. Yamasaki, Concatenate codes, save qubits, *npj Quantum Inf.* **11**, 88 (2025).
- [10] A. R. Calderbank and P. W. Shor, Good quantum error-correcting codes exist, *Phys. Rev. A* **54**, 1098 (1996).
- [11] A. M. Steane, Simple quantum error-correcting codes, *Phys. Rev. A* **54**, 4741 (1996).
- [12] P. Liu, M. Xu, H. Zhou, H. Wang, U. A. Acar, and Y. Shi, ConiQ: Enabling concatenated quantum error correction on neutral atom arrays, [arXiv:2508.05779 \[cs.AR\]](https://arxiv.org/abs/2508.05779).
- [13] R. Rines *et al.*, Demonstration of a logical architecture uniting motion and in-place entanglement: Shor's algorithm, constant-depth CNOT ladder, and many-hypercube code, [arXiv:2509.13247 \[quant-ph\]](https://arxiv.org/abs/2509.13247).
- [14] P. W. Shor, Scheme for reducing decoherence in quantum computer memory, *Phys. Rev. A* **52**, R2493 (1995).
- [15] E. Knill, R. Laflamme, and L. Viola, Theory of quantum error correction for general noise, *Phys. Rev. Lett.* **84**, 2525 (2000).
- [16] D. Poulin, Stabilizer formalism for operator quantum error correction, *Phys. Rev. Lett.* **95**, 230504 (2005).
- [17] D. Bacon, Operator quantum error-correcting subsystems for self-correcting quantum memories, *Phys. Rev. A* **73**, 012340 (2006).
- [18] M. B. Hastings and J. Haah, Dynamically generated logical qubits, *Quantum* **5**, 564 (2021).
- [19] J. Haah and M. B. Hastings, Boundaries for the honeycomb code, *Quantum* **6**, 693 (2022).
- [20] C. Gidney, M. Newman, A. Fowler, and M. Broughton, A fault-tolerant honeycomb memory, *Quantum* **5**, 605 (2021).
- [21] C. Gidney, M. Newman, and M. McEwen, Benchmarking the planar honeycomb code, *Quantum* **6**, 813 (2022).
- [22] A. Townsend-Teague, J. Magdalena de la Fuente, and M. Kesselring, Floquetifying the colour code, *Electron. Proc. Theor. Comput. Sci.* **384**, 265 (2023).
- [23] H. Bombin, D. Litinski, N. Nickerson, F. Pastawski, and S. Roberts, Unifying flavors of fault tolerance with the ZX calculus, *Quantum* **8**, 1379 (2024).
- [24] M. Davydova, N. Tantivasadakarn, and S. Balasubramanian, Floquet codes without parent subsystem codes, *PRX Quantum* **4**, 020341 (2023).
- [25] A. Dua, N. Tantivasadakarn, J. Sullivan, and T. D. Ellison, Engineering 3D floquet codes by rewinding, *PRX Quantum* **5**, 020305 (2024).
- [26] M. S. Alam and E. Rieffel, Dynamical logical qubits in the Bacon-Shor code, *Phys. Rev. Lett.* **112**, 022436 (2024).
- [27] L. Vaidman, L. Goldenberg, and S. Wiesner, Error prevention scheme with four particles, *Phys. Rev. A* **54**, R1745 (1996).
- [28] M. Grassl, T. Beth, and T. Pellizzari, Codes for the quantum erasure channel, *Phys. Rev. A* **56**, 33 (1997).
- [29] D. Poulin, Optimal and efficient decoding of concatenated quantum block codes, *Phys. Rev. A* **74**, 052333 (2006).
- [30] N. Maskara, A. Kubica, and T. Jochym-O'Connor, Advantages of versatile neural-network decoding for topological codes, *Phys. Rev. A* **99**, 052351 (2019).
- [31] P. Baireuther, M. D. Caio, B. Criger, C. W. J. Beenakker, and T. E. O'Brien, Neural network decoder for topological color codes with circuit level noise, *New J. Phys.* **21**, 013003 (2019).
- [32] P. Panteleev and G. Kalachev, Degenerate quantum LDPC codes with good finite length performance, *Quantum* **5**, 585 (2021).
- [33] E. Knill, Quantum computing with realistically noisy devices, *Nature* **434**, 39 (2005).
- [34] B. W. Reichardt, Fault-tolerant quantum error correction for Steane's seven-qubit color code with few or no extra qubits, *Quantum Sci. Technol.* **6**, 015007 (2020).
- [35] C. Gidney, Stim: A fast stabilizer circuit simulator, *Quantum* **5**, 497 (2021).
- [36] J. Roffe, D. R. White, S. Burton, and E. Campbell, Decoding across the quantum low-density parity-check code landscape, *Phys. Rev. Res.* **2**, 043423 (2020).
- [37] J. Roffe, LDPC: Python tools for low density parity check codes (2022). <https://pypi.org/project/ldpc/>.
- [38] S. Varsamopoulos, B. Criger, and K. Bertels, Decoding small surface codes with feedforward neural networks, *Quantum Sci. Technol.* **3**, 015004 (2017).
- [39] P. Baireuther, T. E. O'Brien, B. Tarasinski, and C. W. J. Beenakker, Machine-learning-assisted correction of correlated qubit errors in a topological code, *Quantum* **2**, 48 (2018).
- [40] C. Chamberland and P. Ronagh, Deep neural decoders for near term fault-tolerant experiments, *Quantum Sci. Technol.* **3**, 044002 (2018).
- [41] S. Varsamopoulos, K. Bertels, and C. G. Almudever, Comparing neural network based decoders for the surface code, *IEEE Trans. Comput.* **69**, 300 (2020).
- [42] Within the number of samples used for training: NNs with more nodes and layers do not necessarily perform better and sometimes perform worse at level 4.