nature physics

Article

https://doi.org/10.1038/s41567-023-02325-8

Time-Efficient Constant-Space-Overhead Fault-Tolerant Quantum Computation

In the format provided by the authors and unedited

Supplemental Information on "Time-Efficient Constant-Space-Overhead Fault-Tolerant Quantum Computation" is organized as follows. In Sec. A, we present the setting of fault-tolerant quantum computation (FTQC). In Sec. B, the definition of quantum codes used in our fault-tolerant protocol and the proof of the non-vanishing rate of our quantum code are presented in detail. In Sec. C, we explain how our fault-tolerant protocol compiles the original circuit into the fault-tolerant circuit using gadgets. In Sec. D, we show the construction of the gadgets. In Sec. E, we prove the existence of a threshold in our protocol. In Sec. F, we analyze the space and time overheads of our protocol.

Appendix A: Setting

We present the setting of fault-tolerant quantum computation (FTQC) in our work. Quantum computation aims to solve a family of computational problems using a quantum circuit. We call this circuit the original circuit [1]. A classical input of the computational problems is represented by an M-bit string

$$(i_1, \ldots, i_M) \in \{0,1\}^M. \tag{A1}$$

The original circuit is composed of the following operations: preparations, gates, measurements, and waits performing the identity gate $\mathbb{1}$. Each operation in a circuit is called a location. The width and the depth of the original circuit are denoted by $W(M)$ and $D(M)$, respectively. The original circuit is assumed to have a polynomial size in M; that is, $W(M)$ and $D(M)$ are bounded by

$$W(M) = O(\mathrm{poly}(M)),\ W(M) \geqq M, \tag{A2}$$
$$D(M) = O(\mathrm{poly}(M)),\ D(M) \geqq 0. \tag{A3}$$

The original circuit starts with the preparation of all the $W(M)$ qubits in $|0\rangle^{\otimes W(M)}$. Then, we input the M-bit string $(i_1, \ldots, i_M)$ in (A1) to the original circuit, using an M-qubit input state prepared at the initial part of the original circuit as

$$\bigotimes_{m=1}^{M} |i_m\rangle = \left(\bigotimes_{m=1}^{M} X^{i_m}\right) |0\rangle^{\otimes M}, \tag{A4}$$

where $X^0 = \mathbb{1}$. The part of the original circuit for preparing (A4) is called the *input part*. The original circuit holds $(\bigotimes_{m=1}^{M} |i_m\rangle) \otimes |0\rangle^{\otimes (W(M)-M)}$ at the end of the input part. After the input part, we apply gates to the qubits. We write the original circuit in terms of a gate set of Clifford gates X, Y, Z, H, S, CNOT, and CZ, and non-Clifford gates $R_y(\pm\pi/4)$ [2]. The gates in the original circuit can be performed on all the qubits in parallel, within the constraint that at most one gate can be performed on each qubit at each time step; that is, the depth of the original circuit refers to the number of time steps under this constraint. Two-qubit gates in the original circuit may be performed on arbitrary pairs of the qubits. Finally, we perform measurements of all the $W(M)$ qubits in the Z basis, which achieves sampling of an $W(M)$-bit string that is returned as a result of quantum computation. In our setting, the original circuit does not include measurements in the middle of computation and thus does not have feed-forward operations conditioned on measurement outcomes; that is, the width of the original circuit (i.e., the number of qubits in the original circuit) remains constant for each time step. Except for the input part, the original circuit is fixed, i.e., is independent of $(i_1, \ldots, i_M)$ and depends only on M. Note that in this setting, we can perform computation controlled by $(i_1, \ldots, i_M)$ with controlled gates, using the initial state (A4) for the state of the control qubits. We will show an example of original circuits in Fig. C.1 of Sec. C.

The goal of FTQC is to simulate the original circuit using a circuit on physical qubits that may suffer from errors. Prior to starting the execution of quantum computation, we have the $O(\mathrm{poly}(M))$-size classical description of the original circuit. For FTQC, we compile the original circuit into a circuit on physical qubits, which we call a *fault-tolerant circuit*. The fault-tolerant circuit is composed of the following operations: preparation of a physical qubit in $|0\rangle$, Clifford gates X, Y, Z, H, S, CNOT, and CZ, non-Clifford gates $R_y(\pm\pi/4)$, measurement of each physical qubit in the Z basis, and wait, i.e., performing the identity gate $\mathbb{1}$. Our model is as follows.

1. **Local stochastic error model**: We use a conventional but sufficiently general error model, a *local stochastic error model* [1, 3]. We say that a circuit undergoes a local statistical error model if the faults occurring in the circuit satisfy the following [1]: (i) a set S of faulty locations in the circuit is chosen randomly with probability $p(S)$, and errors occur at the locations in S in such a way that the operations at the locations in S are replaced with an arbitrary quantum channel $\mathcal{E}$ that is consistent with the causal order of the locations in S (i.e., $\mathcal{E}$ may be adversarial and correlated); (ii) each location i in the circuit has a parameter $p_{0,i}$ such that for any set R of locations, the probability $\Pr\{S \supseteqq R\}$ of having faults at every location in R (i.e., the probability of the randomly chosen set S including R) is at most

$$\prod_{i \in R} p_{0,i}. \tag{A5}$$

   We assume that the fault-tolerant circuit undergoes the local stochastic error model.

2. **Parallel quantum operation**: The operations on physical qubits can be performed in parallel, where each qubit is involved in at most one operation at a time.

3. **Faultless classical computation**: Conditioned on the outcome of measurements on physical qubits, we can perform classical computation to change the subsequent operations on the qubits adaptively. For simplicity of analysis of overhead, we assume that classical computation is faultless. Note that our threshold analysis in Sec. E will prove that our fault-tolerant protocol has a threshold even with a polynomially large architectural time overhead, and thus even if classical computation is performed by faulty circuits with the overhead of using a classical error-correcting code [4, 5], a threshold still exists.

4. **Parallel classical computation but nonzero runtime**: Similar to the operations on physical qubits, classical computation is assumed to be performed in parallel so that the depth of classical circuits determines the runtime of classical computation. The number of parallel processes of classical computation is limited to the same order as the number of physical qubits up to a quasi-polylogarithmic factor, as given later by (A8). In the analysis of FTQC, it has been conventional to ignore the runtime of classical computation, i.e., to assume that the classical computation runs instantaneously in zero time that never grows on large scales [1, 3]. However, in practice, the required runtime of classical computation in FTQC such as that for the decoder may become longer on larger scales. In view of this practical requirement, we do not ignore the runtime of classical computation; that is, we take into account the runtime of classical computation in our analysis by performing wait operations on physical qubits during the runtime of classical computation. In our setting, errors may also occur on these wait operations according to the local stochastic error model.

5. **Allocation of qubits and bits**: We allocate qubits by the preparation and deallocate the qubits by the measurement. The measurement allocates bits for classical computation to process the outcome of the measurement. After using the measurement outcomes, we discard the outcomes and deallocate the bits. The number of qubits and bits at a time is that which has already been allocated before and is not yet deallocated at the time.

6. **No geometrical constraint**: Following the convention of the previous works [3, 6, 7], we assume that two-qubit gates are applicable to any pair of physical qubits without geometrical constraint as in photonic systems [8–12] and distributed architectures [13–15]. Classical computation can also be performed without such geometrical constraints. This assumption is essential for avoiding polynomially large time overhead in simulating the original circuit; for example, if one could use only 2D local interactions on m qubits aligned on an $O(\sqrt{m}) \times O(\sqrt{m})$ square lattice, a realization of each long-range two-qubit gate would require $O(\sqrt{m})$ operations to mediate, which we can avoid with our assumption. Note that our threshold analysis in Sec. E will prove that our fault-tolerant protocol still has a threshold even with such polynomial time overhead arising from the 2D nearest-neighbor interactions; in such a case, we can incorporate the techniques in Refs. [16–19] for fault-tolerant implementation of the gates within the geometrical constraints.

A fault-tolerant protocol aims to simulate the $W(M)$-qubit $D(M)$-depth original circuit within any given target error $\epsilon > 0$, i.e., to output an $W(M)$-bit string sampled from a probability distribution close to the output probability distribution of the original circuit within error in the total variational distance at most ϵ. To achieve this simulation, the fault-tolerant protocol replaces the qubits in the original circuit with logical qubits of a quantum error-correcting code, to obtain the fault-tolerant circuit using multiple physical qubits per logical qubit. Let $W_{\mathrm{FT}}(M)$ denote the maximum of the total number of physical qubits allocated at a time, where the maximum is taken over all the time steps in the fault-tolerant circuit. The space overhead refers to $W_{\mathrm{FT}}(M)$ divided by the number $W(M)$ of qubits in the original circuit, i.e.,

$$\frac{W_{\mathrm{FT}}(M)}{W(M)}. \tag{A6}$$

Apart from the physical qubits, for classical computation, we allow the fault-tolerant protocol to use a classical memory with a size of

$$O(\mathrm{poly}(M)) \text{ bits}; \tag{A7}$$

after all, just to store the classical description of the original circuit, we may need to use $O(\mathrm{poly}(M))$ bits. On the other hand, we limit the number of parallel processes of classical computation to be of the same order as the number $W(M)$ of qubits up to a quasi-polylogarithmic factor, i.e.,

$$W(M) \times \exp(O(\mathrm{polylog}(\log(M)))). \tag{A8}$$

The time overhead refers to the depth $D_{\mathrm{FT}}(M)$ of the fault-tolerant circuit (including wait operations to wait for classical computation) divided by $D(M)$ of the original circuit, i.e.,

$$\frac{D_{\mathrm{FT}}(M)}{D(M)}. \tag{A9}$$

Note that, due to the limitation (A7) of the number of bits, it is prohibited to use an exponentially large number of parallel processes of classical computation to classically simulate the original circuit in a polynomial time in

terms of the circuit depth. For fixed ϵ, the fault-tolerant protocol is said to achieve a constant space overhead if the space overhead is $O(1)$ as $M \to \infty$.

The classical process for converting the original circuit into a fault-tolerant circuit is called compilation. We perform the compilation prior to starting the execution of quantum computation, i.e., prior to knowing the input $(i_1, \dots, i_M)$ in (A1) and allocating physical qubits. By convention, we do not include the runtime of compilation in the time overhead. Note that this setting may potentially justify solving an NP-hard problem of circuit optimization heuristically during compilation, but our fault-tolerant protocol will not require any process to solve such an NP-hard problem in our compilation; in particular, we will clarify in Secs. C and D that the compilation can be performed feasibly using techniques for converting the stabilizer circuits. Also, due to the limitation (A7) of the number of bits, it is prohibited to store the result of classical simulation of the original circuit for all possible inputs (A1) by performing an exponentially long-time classical computation during the compilation.

Appendix B: Concatenated code constructed from a sequence of quantum Hamming codes

In this section, we summarize the notations on quantum codes and define the quantum error-correcting code used in our fault-tolerant protocol. For details of quantum codes, see also Refs. [1, 2, 20–23].

A quantum code $\mathcal{Q}$ on N physical qubits is a subspace of $(\mathbb{C}^2)^{\otimes N}$, where $\mathbb{C}^2 = \mathrm{span}\{|0\rangle, |1\rangle\}$ represents a qubit. A stabilizer code $\mathcal{Q}(\mathcal{S})$ is a quantum code represented by a stabilizer $\mathcal{S} \subset \mathcal{P}_N$, i.e., an Abelian subgroup of the Pauli group $\mathcal{P}_N$ on N qubits generated by $\{\mathrm{i}\mathbb{1}, X_1, Z_1, \dots, X_N, Z_N\}$ with $-\mathbb{1} \notin \mathcal{S}$, where X_n and Z_n for each $n \in \{1, \dots, N\}$ are Pauli-X and Z operators, respectively, acting on the nth qubit. A centralizer $C(\mathcal{S})$ of $\mathcal{S}$ is the set of Pauli operators in $\mathcal{P}_N$ that commute with all the elements in $\mathcal{S}$. Suppose that $\mathcal{S}$ is generated by $N-K$ independent generators. The code space $\mathcal{Q}(\mathcal{S})$ is the 2^K-dimensional subspace of N qubits spanned by all N-qubit states invariant under the action of $\mathcal{S}$, which defines K logical qubits. Logical operators are elements in $C(\mathcal{S})/\mathcal{S}$, which is isomorphic to the Pauli group $\mathcal{P}_K$ on K qubits [1]. In particular, for each $k \in \{1, \dots, K\}$, the logical X and Z operators of the kth logical qubit of $\mathcal{Q}(\mathcal{S})$ are defined as the elements in $C(\mathcal{S})/\mathcal{S}$ that are identified with X_k and Z_k, respectively, in $\mathcal{P}_K$ acting on the kth qubit, under this isomorphism. The rate of $\mathcal{Q}(\mathcal{S})$ is K/N. The distance D of $\mathcal{Q}(\mathcal{S})$ is the minimal weight of nontrivial logical operators, where the weight of an N-qubit operator is the number of qubits on which the operator acts nontrivially (rather than $\mathbb{1}$). A stabilizer code with N physical qubits, K logical qubits, and distance D is written as an $[[N, K, D]]$ code in double square brackets. We may call N the size of the code block.

A Calderbank-Shor-Steane (CSS) code provides a way of constructing a class of stabilizer codes from two classical linear codes. A classical linear code $\mathcal{C}$ of block length N is a linear subspace of $\mathbb{F}_2^N$, where $\mathbb{F}_2 = \{0, 1\}$ is the finite field of order 2 representing a bit [24]. The linear code $\mathcal{C}$ is characterized by an $(N-K) \times N$ parity-check matrix H such that the inner product of any codeword $v \in \mathcal{C}$ and any row of H is zero, i.e., $\mathcal{C} = \{v \in \mathbb{F}_2^N : Hv^T = 0\}$, where the codeword v of $\mathcal{C}$ is represented as a row vector, and thus the transpose v^T of v is a column vector. Each row of H is used as a parity check to detect and correct errors. The dimension of $\mathcal{C}$ is $K := \log_2(\dim \mathcal{C})$ in bits. The rate of $\mathcal{C}$ is K/N. The distance D of $\mathcal{C}$ is the minimal Hamming distance between any distinct codewords $u, v \in \mathcal{C}$, where the Hamming distance of u and v is the weight, i.e., the number of 1s in the N elements, of $u + v$. A classical linear code of block length N, dimension K, and distance D is written as an $[N, K, D]$ code in single square brackets. To obtain a CSS code, an $[N, K_Z, D_Z]$ code $\mathcal{C}_Z$ and an $[N, K_X, D_X]$ code $\mathcal{C}_X$ of the same block length N satisfying $\mathcal{C}_X^\perp \subseteqq \mathcal{C}_Z$ are used, where $\mathcal{C}_X^\perp$ is the dual code of $\mathcal{C}_X$ spanned by $(N - K_X)$ rows of the parity-check matrix of $\mathcal{C}_X$, i.e., $\mathcal{C}_X^\perp = \{u \in \mathbb{F}_2^N : \forall v \in \mathcal{C}_X, \langle u \,|\, v\rangle = 0\}$. Let H_Z and H_X denote the parity-check matrices of $\mathcal{C}_Z$ and $\mathcal{C}_X$, respectively; then, the condition $\mathcal{C}_X^\perp \subseteqq \mathcal{C}_Z$ is equivalent to $H_Z H_X^T = 0$, where H_X^T is the transpose of H_X. The CSS code of $\mathcal{C}_Z$ over $\mathcal{C}_X^\perp$ [2] is a quantum code denoted by $\mathcal{Q}_{\mathrm{CSS}}(\mathcal{C}_Z, \mathcal{C}_X) := \mathrm{span}\{\sum_{w \in \mathcal{C}_X^\perp} |u + w\rangle : u \in \mathcal{C}_Z\}$. The code $\mathcal{Q}_{\mathrm{CSS}}(\mathcal{C}_Z, \mathcal{C}_X)$ is a class of stabilizer codes with stabilizer generators given only in Xs or only in Zs. In particular, for any row vector $(b_1, \dots, b_N) \in \mathbb{F}_2^N$ given by a row of H_Z, the stabilizer of $\mathcal{Q}_{\mathrm{CSS}}(\mathcal{C}_Z, \mathcal{C}_X)$ has a Z generator $\bigotimes_{n=1}^N Z_n^{b_n}$, where $Z^1 = Z$ and $Z^0 = \mathbb{1}$. Also, for any row $(b_1, \dots, b_N) \in \mathbb{F}_2^N$ of H_X, the stabilizer of $\mathcal{Q}_{\mathrm{CSS}}(\mathcal{C}_1, \mathcal{C}_2)$ has an X generator $\bigotimes_{n=1}^N X_n^{b_n}$, where $X^1 = X$ and $X^0 = \mathbb{1}$. As a whole, $\mathcal{Q}_{\mathrm{CSS}}(\mathcal{C}_Z, \mathcal{C}_X)$ has $(N - K_Z)$ stabilizer generators in Z and $(N - K_X)$ stabilizer generators in X. The condition $\mathcal{C}_X^\perp \subseteqq \mathcal{C}_Z$, i.e., $H_Z H_X^T = 0$, requires that each pair of Z and X generators should commute with each other, so that the stabilizer of $\mathcal{Q}_{\mathrm{CSS}}(\mathcal{C}_Z, \mathcal{C}_X)$ should be Abelian. The CSS code $\mathcal{Q}_{\mathrm{CSS}}(\mathcal{C}_Z, \mathcal{C}_X)$ is an $[[N, K_Z + K_X - N, D]]$ stabilizer code, where the distance is lower bounded by $D \geqq \min\{D_Z, D_X\}$.

Hamming codes $(\mathcal{C}_r : r = 2, 3, \dots)$ are a family of linear classical codes such that the parity-check matrix

$$H_r \tag{B1}$$

of $\mathcal{C}_r$ is an $r \times N_r$ matrix where $N_r := 2^r - 1$, and the ith row for each $i \in \{1, \dots, r\}$ has 1s between $(2^{i-1} + 2^i j)$th–$(2^i - 1 + 2^i j)$th columns for all $j \in \{0, 1, \dots\}$ and 0s otherwise [25]. For example, Table I gives the ith row $(b_{i,1}, b_{i,2}, \dots, b_{i,N_r})$ of H_r. The code $\mathcal{C}_r$ has block length N_r, dimension $N_r - r$, and distance 3; that is, $\mathcal{C}_r$ is an $[N_r, N_r - r, 3]$ code.

Quantum Hamming codes $(\mathcal{Q}_r : r = 3, 4, \dots)$ are defined as CSS codes of $\mathcal{C}_r$ over $\mathcal{C}_r^\perp$ [26]. Since $\mathcal{C}_r$ is weakly

TABLE I. The parity-check matrix H_r of the Hamming codes $\mathcal{C}_r$ with $r \in \{2,3,\dots\}$. The ith row of H_r is denoted by $(b_{i,1}, b_{i,2}, \dots, b_{i,N_r})$ and given in the table, where $i \in \{1,\dots,r\}$ and $N_r = 2^r - 1$.

i	$b_{i,1}$	$b_{i,2}$	$b_{i,3}$	$b_{i,4}$	$b_{i,5}$	$b_{i,6}$	$b_{i,7}$	$b_{i,8}$	$b_{i,9}$	$b_{i,10}$	$b_{i,11}$	$b_{i,12}$	$b_{i,13}$	$b_{i,14}$	$b_{i,15}$	$b_{i,16}$	$b_{i,17}$	$b_{i,18}$	$b_{i,19}$	$\cdots$
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
2	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	
3	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	
4	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
$\vdots$																				

self-dual, i.e., $H_r H_r^T = 0$, the condition for CSS codes is satisfied. For each $i \in \{1,\dots,r\}$ to specify the ith row $(b_{i,1},\dots,b_{i,N_r}) \in \{0,1\}^{N_r}$ of the parity-check matrix H_r of $\mathcal{C}_r$, the stabilizer of $\mathcal{Q}_r$ has a Z generator and an X generator given respectively by

$$\bigotimes_{n=1}^{N_r} Z_n^{b_{i,n}}, \tag{B2}$$

$$\bigotimes_{n=1}^{N_r} X_n^{b_{i,n}}. \tag{B3}$$

As a whole, the stabilizer of $\mathcal{Q}_r$ has r Z generators and r X generators. A code block of $\mathcal{Q}_r$ has

$$N_r = 2^r - 1 \text{ physical qubits}, \tag{B4}$$

$$K_r \coloneqq N_r - 2r \text{ logical qubits}, \tag{B5}$$

and distance 3; that is, $\mathcal{Q}_r$ is an $[[N_r, K_r, 3]]$ code. For example, $\mathcal{Q}_3$ is a $[[7,1,3]]$ code also known as Steane's 7-qubit code [27, 28], $\mathcal{Q}_4$ is $[[15,7,3]]$, $\mathcal{Q}_5$ is $[[31,21,3]]$, $\mathcal{Q}_6$ is $[[63,51,3]]$, $\mathcal{Q}_7$ is $[[127,113,3]]$, $\mathcal{Q}_8$ is $[[255,239,3]]$, and $\mathcal{Q}_9$ is $[[511,493,3]]$. The rate of $\mathcal{Q}_r$ converges to

$$\lim_{r\to\infty} \frac{K_r}{N_r} = 1, \tag{B6}$$

which is necessary for constructing a family of concatenated codes with a non-vanishing rate in the limit of large concatenation levels. For each $k \in \{1,\dots,K_r\}$, the kth logical qubit of $\mathcal{Q}_r$ has the logical Z and X operators given respectively by

$$\bigotimes_{n=1}^{N_r} Z_n^{b_n^{(k)}},$$

$$\bigotimes_{n=1}^{N_r} X_n^{b_n^{(k)}}, \tag{B7}$$

in terms of the same bit string $(b_1^{(k)},\dots,b_{N_r}^{(k)}) \in \{0,1\}^{N_r}$ representing the kth logical qubit, which can be calculated by techniques in Refs. [29, 30]. Although the choice of this bit string $(b_1^{(k)},\dots,b_{N_r}^{(k)})$ for each r and k is not unique, it is assumed here and henceforth that a fixed choice of the bit string is adopted. We let

$$G_r \tag{B8}$$

denote a $K_r \times N_r$ matrix where the kth row for $k \in \{1,\dots,K_r\}$ is $(b_1^{(k)},\dots,b_{N_r}^{(k)})$. Note that we have $G_r H_r^T = 0$ as with a generator matrix G and a parity-check matrix H of a classical linear code satisfying $GH^T = 0$.

Note that, for a family of CSS codes, each code in the family is called a quantum low-density parity-check (LDPC) code if the parity-check matrices H_Z and H_X of $\mathcal{C}_Z$ and $\mathcal{C}_X$ for the CSS codes in the family have $O(1)$ nonzero elements in each column and row as the code size increases; that is, the Z and X stabilizer generators of the quantum LDPC codes have only $O(1)$ weights, and only a constant number $O(1)$ of stabilizer generators act nontrivially on each physical qubit of the quantum LDPC codes. By definition, quantum Hamming codes are not quantum LDPC codes.

The following observations are essential for implementing logical gates [1].

- Since $\mathcal{Q}_r$ is a CSS code obtained from a weakly self-dual linear code $\mathcal{C}_r$, transversal CNOT and CZ gates between all the physical qubits between two code blocks of $\mathcal{Q}_r$ implement logical CNOT and CZ gates between all the logical qubits between the code blocks. Also, transversal H gates acting on all the N_r physical qubits of $\mathcal{Q}_r$, i.e., $H^{\otimes N_r}$, implement a logical operator $H^{\otimes K_r}$ on all the K_r logical qubits of $\mathcal{Q}_r$.

- Since $\mathcal{Q}_r$ is a stabilizer code, any logical Pauli operation in the Pauli group $\mathcal{P}_{K_r}$ corresponds to a physical Pauli operation in the Pauli group $\mathcal{P}_{N_r}$, and is thus implemented by applying a suitable Pauli gate (X, Z, Y, or $\mathbb{1}$) on each of the N_r physical qubits.

For example, consider the $[[7,1,3]]$ code $\mathcal{Q}_3$. The stabilizer is generated by

$$Z \otimes \mathbb{1} \otimes Z \otimes \mathbb{1} \otimes Z \otimes \mathbb{1} \otimes Z, \tag{B9}$$

$$X \otimes \mathbb{1} \otimes X \otimes \mathbb{1} \otimes X \otimes \mathbb{1} \otimes X, \tag{B10}$$

$$\mathbb{1} \otimes Z \otimes Z \otimes \mathbb{1} \otimes \mathbb{1} \otimes Z \otimes Z, \tag{B11}$$

$$\mathbb{1} \otimes X \otimes X \otimes \mathbb{1} \otimes \mathbb{1} \otimes X \otimes X, \tag{B12}$$

$$\mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes Z \otimes Z \otimes Z \otimes Z, \tag{B13}$$

$$\mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes X \otimes X \otimes X \otimes X. \tag{B14}$$

The logical Z and X operators on the logical qubit are, respectively,

$$Z \otimes Z \otimes Z \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1}, \tag{B15}$$
$$X \otimes X \otimes X \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1}. \tag{B16}$$

The logical H on the logical qubit is given by

$$H \otimes H \otimes H \otimes H \otimes H \otimes H \otimes H. \tag{B17}$$

Here the construction of the concatenated code $\mathcal{Q}^{(L)}$ is described in full detail. For concatenation, we use a sequence of quantum Hamming codes $(\mathcal{Q}_{r_l} : 1, 2, \ldots, L)$, where we choose the parameter r_l as

$$r_l := l + 2. \tag{B18}$$

Note that, due to (B4) and (B5), this choice of r_l leads to

$$N_{r_l} = \exp(O(l)), \tag{B19}$$
$$K_{r_l} = \exp(O(l)), \tag{B20}$$

as $l \to \infty$. For $l \in \{L, L-1, \ldots, 1, 0\}$, the construction here introduces a collection of $K^{(l)}$ qubits called a *level-l register*, where $K^{(l)}$ is explicitly given by

$$K^{(l)} := \prod_{l'=1}^{l} K_{r_{l'}} = \prod_{l'=1}^{l} (2^{r_{l'}} - 2r_{l'} - 1) \tag{B21}$$

and $K^{(0)} := 1$. Every qubit in a level-l register is distinguished by a label

$$k^{(l)} \in \{1, \ldots, K^{(l)}\}. \tag{B22}$$

The code $\mathcal{Q}^{(L)}$ is constructed recursively for $l = L, L-1, \ldots, 1$ by encoding the $K^{(l)} = K_{r_l} \times K^{(l-1)}$ qubits in a level-l register into $N_{r_l} \times K^{(l-1)}$ qubits in a set of N_{r_l} level-$(l-1)$ registers, using $K^{(l-1)}$ blocks of the quantum Hamming code $\mathcal{Q}_{r_l}$. The level-L register represents the set of logical qubits of $\mathcal{Q}^{(L)}$, and a level-0 register (or a level-0 qubit) refers to a physical qubit of $\mathcal{Q}^{(L)}$. Encoding into lower levels of qubits involves all the qubits in a register, and consequently, the qubits in the register may suffer from a common fault at the same time. In contrast, qubits belonging to distinct registers should have independent statistics of their errors if physical errors at level 0 occur independently.

To specify the assignment of each qubit in a level-l register to a code block according to its index $k^{(l)}$, define a map $k^{(l)} \mapsto (k, k^{(l-1)})$ by the relation

$$k^{(l)} = (k-1)K^{(l-1)} + k^{(l-1)}, \tag{B23}$$

where $k^{(l-1)} \in \{1, \ldots, K^{(l-1)}\}$ and $k \in \{1, \ldots, K_{r_l}\}$. The $k^{(l)}$th qubit in the level-l register is then assigned as the kth logical qubit of the $k^{(l-1)}$th block of $\mathcal{Q}_{r_l}$. Each code block encodes K_{r_l} logical qubits into N_{r_l}. Hence, the $K^{(l-1)}$ code blocks have $K_{r_l} \times K^{(l-1)} = K^{(l)}$ logical qubits in total to cover all the qubits in a level-l register, which are encoded into $N_{r_l} \times K^{(l-1)}$ qubits in the set of N_{r_l} level-$(l-1)$ registers as a whole. These qubits in the set of N_{r_l} level-$(l-1)$ registers are naturally indexed by

$$(n, k^{(l-1)}) \tag{B24}$$

with $n \in \{1, \ldots, N_{r_l}\}$, indicating the nth physical qubit of the $k^{(l-1)}$th block of $\mathcal{Q}_{r_l}$. Note that the indexes of the logical and physical qubits of $\mathcal{Q}_{r_l}$ used here should follow (B7) with a fixed choice of the bit strings.

Finally, these $N_{r_l} K^{(l-1)}$ qubits are sorted into N_{r_l} level-$(l-1)$ registers. This is simply done by assigning the qubit indexed by $(n, k^{(l-1)})$ to the $k^{(l-1)}$th qubit in the nth level-$(l-1)$ register. Now the whole procedure of encoding a level-l register into N_{r_l} level-$(l-1)$ registers is specified, which can be recursively applied to define the concatenated code $\mathcal{Q}^{(L)}$ that encodes $K^{(L)}$ qubits in a level-L register into level-0 physical qubits. In the above rule of assignment, every physical qubit of a code $\mathcal{Q}_{r_l}$ is assigned to a distinct level-$(l-1)$ register. This also means that every qubit in a level-$(l-1)$ register is assigned to a distinct block of $\mathcal{Q}_{r_l}$. As mentioned above, qubits in the same register may suffer from errors simultaneously. The construction here is thus intended to work analogously to interleaving techniques widely used for burst error correction.

The number $N^{(L)}$ of physical qubits of $\mathcal{Q}^{(L)}$ is computed from the fact that a level-l register is encoded into N_{r_l} level-$(l-1)$ registers, and that a level-0 register is a single physical qubit by definition; thus, it holds that

$$N^{(L)} = \prod_{l=1}^{L} N_{r_l} = \prod_{l=1}^{L} (2^{r_l} - 1). \tag{B25}$$

The number of logical qubits of $\mathcal{Q}^{(L)}$ is $K^{(L)}$ by definition, i.e.,

$$K^{(L)} = \prod_{l=1}^{L} K_{r_l} = \prod_{l=1}^{L} (2^{r_l} - 2r_l - 1). \tag{B26}$$

Since quantum Hamming codes are distance-3 codes, the distance of $\mathcal{Q}^{(L)}$ obtained by concatenating the quantum Hamming codes L times is 3^L. Therefore, $\mathcal{Q}^{(L)}$ is a $[[N^{(L)}, K^{(L)}, 3^L]]$ code. Note that the ability of our code to suppress errors is not fully characterized by the code distance since our protocol suppresses error by increasing the concatenation level L, unlike quantum LDPC codes increasing distance for error suppression. For the choice of (B18), we have

$$N^{(L)} = \exp(O(L^2)), \tag{B27}$$
$$K^{(L)} = \exp(O(L^2)). \tag{B28}$$

We here prove that $\mathcal{Q}^{(L)}$ has an asymptotically nonvanishing rate. Let

$$R(L) := \frac{K^{(L)}}{N^{(L)}} \tag{B29}$$

denote the rate of $\mathcal{Q}^{(L)}$, where $K^{(L)}$ and $N^{(L)}$ are given by (B26) and (B25), respectively. Then, it suffices to derive a lower bound of

$$\frac{K^{(L)}}{N^{(L)}} = \prod_{l=1}^{L} \frac{K_{r_l}}{N_{r_l}} \tag{B30}$$

$$= \prod_{l=1}^{L} \frac{2^{r_l} - 2r_l - 1}{2^{r_l} - 1} \tag{B31}$$

$$= \prod_{l=1}^{L} \left(1 - \frac{2(l+2)}{2^{l+2} - 1}\right). \tag{B32}$$

Since $K^{(L)}/N^{(L)}$ monotonically decreases as L increases, it holds for any L that

$$\frac{K^{(L)}}{N^{(L)}} \geqq \prod_{l=1}^{\infty} \frac{K_{r_l}}{N_{r_l}} = \prod_{l=1}^{\infty} \left(1 - \frac{2(l+2)}{2^{l+2} - 1}\right). \tag{B33}$$

Here, an infinite sum

$$\sum_{l=1}^{\infty} \left| \frac{N_{r_l} - K_{r_l}}{N_{r_l}} \right| = \sum_{l=1}^{\infty} \left| \frac{2(l+2)}{2^{l+2} - 1} \right| \tag{B34}$$

converges to a finite constant, which proves that the infinite product on the right-hand side of (B33) converges to a positive constant [31]. Therefore, for any L, the rate $R(L)$ of $\mathcal{Q}^{(L)}$ is lower bounded by a positive constant, i.e.,

$$R(L) = \frac{K^{(L)}}{N^{(L)}} \geqq \lim_{l\to\infty} \frac{K^{(l)}}{N^{(l)}} = \frac{1}{\eta_\infty} > 0, \tag{B35}$$

where η_∞ is given by

$$\eta_\infty \coloneqq \lim_{l\to\infty} \frac{N^{(l)}}{K^{(l)}}, \tag{B36}$$

and called an asymptotic overhead factor. The value of the asymptotic overhead factor η_∞ of $\mathcal{Q}^{(L)}$ can be obtained by evaluating (B32) numerically. In particular, the numerical evaluation yields

$$\eta_\infty < 36, \tag{B37}$$

which contrasts with the fact that such overhead factors of conventional concatenated and topological codes would diverge to infinity.

We remark that the above proof of non-vanishing rate for concatenated codes is also applicable to other sequences of codes as long as the sequence of the codes satisfies the condition in (B34). In particular, the above argument leads to the following theorem.

Theorem 1 (Concatenated code with non-vanishing rate)**.** For any sequence of $[[n_l, k_l, d]]$ quantum codes ($l \in \{1, \ldots, L\}$), the above construction leads to a $[[\prod_{l=1}^{L} n_l, \prod_{l=1}^{L} k_l, d^L]]$ concatenated code, and if it holds that

$$\sum_{l=1}^{\infty} \left| \frac{n_l - k_l}{n_l} \right| < \infty, \tag{B38}$$

then the concatenated code has a non-vanishing rate

$$\frac{\prod_{l=1}^{L} k_l}{\prod_{l=1}^{L} n_l} > \prod_{l=1}^{\infty} \frac{k_l}{n_l} > 0. \tag{B39}$$

For example, as we will discuss later in (F41), concatenation of quantum Hamming codes with parameter $r_l = \lceil \beta \log_2(l) \rceil$ for $\beta > 1$ in place of (B18) still achieves a non-vanishing rate and can be used for constant-space-overhead FTQC. In the same way, concatenation of a growing sequence of, e.g., $[[2^{r_l} - 1, 2^{r_l} - 1 - 2tr_l, 2t + 1]]$ quantum Bose-Chaudhuri-Hocquenghem (BCH) codes ($t \geqq 1$) [26] can also provide a family of concatenated codes with non-vanishing rate using the same r_l as (B18). This code family has a larger distance than a quantum Hamming code of the same size and thus may have a potential advantage in faster error suppression as L increases. Nevertheless, our analysis in the following sections focuses on an explicit construction of the fault-tolerant protocol for $\mathcal{Q}^{(L)}$. We leave construction and analysis of general fault-tolerant protocols for other code families for future work, but our explicit construction of the fault-tolerant protocol for $\mathcal{Q}^{(L)}$ clarifies a fundamental design principle for such protocols.

Appendix C: Compilation of original circuit into fault-tolerant circuit

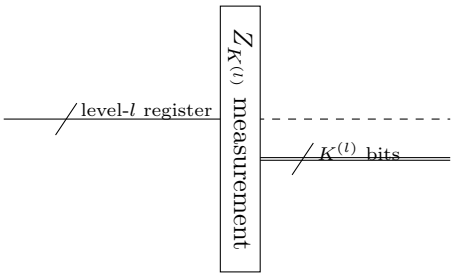
In this section, we present our fault-tolerant protocol that compiles the $W(M)$-qubit $D(M)$-depth original circuit into the fault-tolerant circuit to achieve the target error ϵ. In place of the $W(M)$ qubits of the original circuit, we will use the qubits in level-L registers of the quantum error-correcting code $\mathcal{Q}^{(L)}$, where L is the required concatenation level depending on M and ϵ. In particular, for given M and ϵ, L will be given by (F10) in Sec. E. We here compile the original circuit on the $W(M)$ qubits into a level-L circuit using $O(W(M)/K^{(L)})$ level-L registers. As in the conventional fault-tolerant protocol for concatenated codes in Ref. [1], the actual implementation of the intended level-L circuit by a level-0 circuit, i.e., the fault-tolerant circuit, is derived by recursively constructing the corresponding level-$(l-1)$ circuit from that at level l ($l = L, L-1, \ldots, 1$). But the construction here is different from that of Ref. [1] in that the circuits here are composed of elementary operations acting on registers rather than qubits, and that the procedure of converting a level-l circuit to a level-$(l-1)$ circuit depends on l.

More concretely, for each level $l = 0, \ldots, L$, we will define a set of elementary operations (preparations, gates,

a measurement, and wait) acting on level-l registers. We require that a level-l circuit should be written in terms only of the level-l elementary operations in the set. In a level-l circuit that satisfies this requirement, each of the level-l elementary operations is called a level-l *location*. For each level-l elementary operation, a corresponding level-l *gadget* will be defined, which is a level-$(l-1)$ circuit intended to carry out the logical elementary operation on the encoded level-l registers. We will also define a level-l error-correction gadget, which is a level-$(l-1)$ circuit intended to carry out quantum error correction for an encoded level-l register. To implement some of the gates required for universal quantum computation, our protocol may use gate teleportation that combines multiple level-l elementary operations to implement a gate. For simplicity of presentation, we may represent these multiple level-l elementary operations collectively as a level-l *abbreviation*, which is a level-l circuit intended to carry out the gate on the level-l registers. A level-l circuit that includes level-l abbreviations is identified with that composed only of level-l elementary operations obtained by expanding all the level-l abbreviations. In the following, we list up the set of level-l elementary operations, gadgets, and abbreviations, while the precise definitions will be given later in Sec. D.

We use the following set of level-l elementary operations:

measurement:

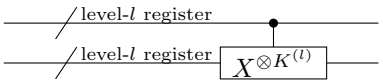


, (C1)

H gate:

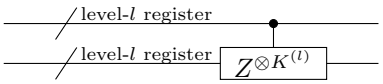
, (C2)

CNOT gate:



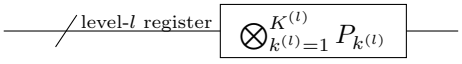
, (C3)

CZ gate:



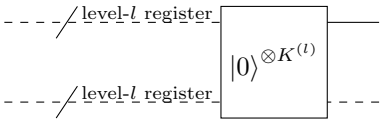
, (C4)

Pauli gate:



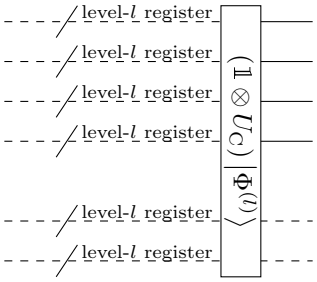
, (C5)

initial-state preparation:



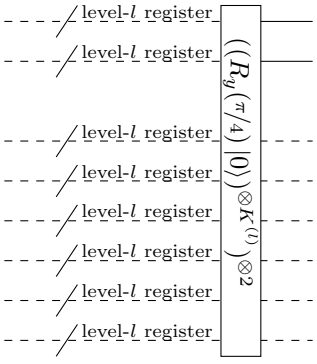
, (C6)

Clifford-state preparation:



, (C7)

magic-state preparation:



, (C8)

wait:

. (C9)

In these notations, a dashed input wire turning into a solid output wire implies the allocation of a level-l register. A solid wire turning into a dashed one implies deallocation. When an input wire and an output wire are both dashed, the corresponding level-l register is used as a workspace in implementing the elementary operation. Double-line wires represent $K^{(l)}$ bits. The level-l measurement operation performs measurements in Z basis $\{|0\rangle, |1\rangle\}$ of all the $K^{(l)}$ qubits in a level-l register, deallocating the level-l register and outputting the $K^{(l)}$-bit measurement outcome, where we let $Z_{K^{(l)}}$ denote a label for representing this measurement. The level-l H-, CNOT-, and CZ-gate operations perform $H^{\otimes K^{(l)}}$, $\mathrm{CNOT}^{\otimes K^{(l)}}$, and $CZ^{\otimes K^{(l)}}$ gates acting on all the qubits in level-l registers. The level-l Pauli-gate operation performs a tensor product of arbitrary Pauli gates

$$\bigotimes_{k^{(l)}=1}^{K^{(l)}} P_{k^{(l)}}, \tag{C10}$$

where $P_{k^{(l)}} \in \{X, Z, Y, \mathbb{1}\}$ is a single-qubit Pauli (or identity) gate acting on the $k^{(l)}$th qubit with labeling (B23), and the global phase is ignored. The level-l initial-

state preparation operation allocates two level-l registers B_1, B_2 and prepares a state $|0\rangle^{\otimes K^{(l)}}$ of $K^{(l)}$ qubits in B_1, followed by deallocating the level-l register B_2 used as a workspace. The level-l Clifford-state preparation operation allocates six level-l registers $B_1, B_2, B_3, B_4, B_5, B_6$ aligned from top to bottom in (C7) and prepares a state $\left(\mathbb{1}^{B_1B_2} \otimes U_{\rm C}^{B_3B_4}\right) |\Phi^{(l)}\rangle^{B_1B_2B_3B_4}$, where $U_{\rm C}^{B_3B_4}$ is an arbitrary Clifford unitary operator acting on qubits in B_3 and B_4,

$$|\Phi^{(l)}\rangle^{B_1B_2B_3B_4} = |\Phi\rangle^{B_1B_3} \otimes |\Phi\rangle^{B_2B_4}, \tag{C11}$$

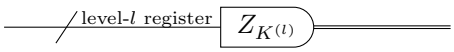
and

$$|\Phi\rangle^{B_jB_{j'}} = \frac{1}{\sqrt{2^{K^{(l)}}}} \sum_{m=0}^{2^{K^{(l)}}-1} |m\rangle^{B_j} \otimes |m\rangle^{B_{j'}}, \quad (j, j') \in \{(1,3),(2,4)\}. \tag{C12}$$

Here, the superscripts represent the registers to which states and operators belong. The two auxiliary level-l registers B_5 and B_6 are deallocated at the end of the implementation. The level-l magic-state preparation operation allocates eight level-l registers $B_1, B_2, B_3, B_4, B_5, B_6, B_7, B_8$ aligned from top to bottom in (C8), prepares a state $(R_y(\pi/4)\,|0\rangle)^{\otimes K^{(l)}}$ of $K^{(l)}$ qubits in each of B_1 and B_2, and deallocates $B_3, B_4, B_5, B_6, B_7, B_8$ at the end of the implementation. To avoid potential confusion, in our presentation, we always write the registers $B_1, B_2, \ldots$ in (C6), (C7), and (C8) from top to bottom in a circuit. However, under our assumption of no geometrical constraint, we can arbitrarily bend, cross, and permute the wires, which we do not consider as elementary operations. The level-l wait operation performs the identity operator on a level-l register, which will be regarded as a special case of Pauli gates henceforth and will not be explained explicitly.

We may omit dashed wires output from the operations while we do not omit those input into the operations. It is then possible to determine the number of omitted output wires by comparing the total numbers of the input and the explicitly shown output wires. For example, the measurement operation (C1) with omitting the output dashed wire may be denoted by

measurement:

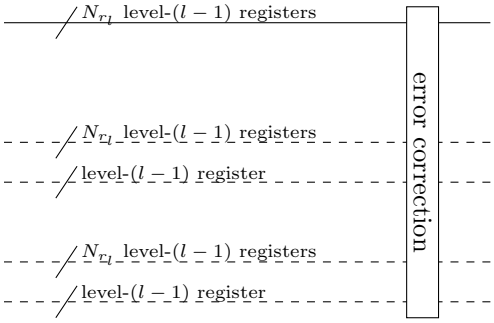


. (C13)

The notations on elementary operations may also be used to show the corresponding gadgets; in particular, whenever a level-l elementary operation is depicted as the one acting on N_{r_l} level-$(l-1)$ registers instead of each level-l register, this notation represents the corresponding level-l gadget.

Apart from these elementary operations and the corresponding gadgets, the level-l error-correction gadget is denoted by

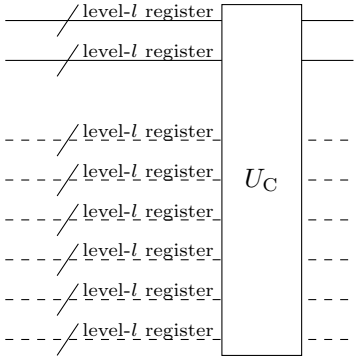
error correction:



. (C14)

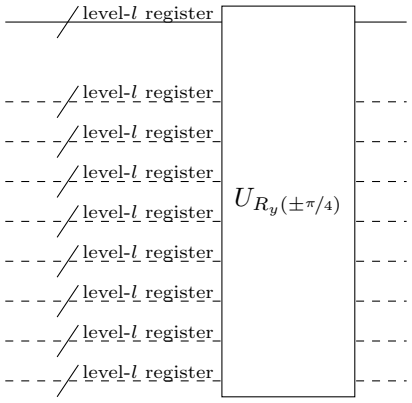
The level-l error-correction gadget performs quantum error correction on an encoded level-l register for the first set of N_{r_l} level-$(l-1)$ registers (solid wire), temporarily using the other $2(N_{r_l}+1)$ level-$(l-1)$ registers (dashed wires). In addition, the following level-l abbreviations are used:

two-register Clifford gate:



, (C15)

$R_y(\pm\pi/4)$ gate:



, (C16)

which are abbreviated notations of combinations of level-l elementary operations with functions described as follows. The level-l two-register Clifford-gate abbreviation applies an arbitrary Clifford unitary $U_{\rm C}$ on the first two level-l registers (solid wires), temporarily using the other six auxiliary level-l registers (dashed wires) during its implementation. The level-l $R_y(\pm\pi/4)$-gate abbreviation applies $U_{R_y(\pm\pi/4)}$ gates on the first level-l register (solid wire), temporarily using the other eight auxiliary

level-l registers (dashed wires) during its implementation, where $U_{R_y(\pm\pi/4)}$ is an arbitrary tensor product of $R_y(\pi/4)$, $R_y(-\pi/4)$, and $\mathbb{1}$. Similar to the elementary operations, the output dashed wires in these notations may be omitted.

Using these elementary operations and abbreviations, we compile the original circuit into a level-L circuit as shown in Fig. C.1. In particular, we use $\lceil W(M)/K^{(L)} \rceil$ level-L registers and represent the $W(M)$ qubits in the original circuit as the qubits in these level-L registers, where

$$\lceil x \rceil \tag{C17}$$

is the ceiling function representing the smallest integer larger than or equal to x. Additionally, for each of these level-L registers, we allocate $8+2=10$ auxiliary level-L registers. Among these ten, eight auxiliary level-L registers are used for the level-L abbreviations and the level-L elementary operations as explained in the following. The other two, which we call level-L *dormant* registers, are never used explicitly in the level-L circuit. The level-L dormant registers are used to secure a workspace for the level-L error-correction gadgets appearing in the level-$(L-1)$ circuit, as will be explained later. To obtain the level-L circuit, we replace the Pauli gates in the input part (A4) of the original circuit with the corresponding level-L Pauli-gate operations, and then replace the Clifford and $R_y(\pm\pi/4)$ gates in the original circuit with the corresponding level-L Clifford- and $R_y(\pm\pi/4)$-gate abbreviations, respectively; in this replacement, the dashed wires in the level-L abbreviations (C15) and (C16) are always selected from the eight out of the ten auxiliary level-L registers added to the level-L registers on which the abbreviations act. We also replace preparations of $|0\rangle$ and measurements in the Z basis in the original circuit with the corresponding level-L preparation and measurement operations, respectively, in the level-L circuit, where each level-L preparation operation uses, as the dashed wire in (C6), one of the eight auxiliary level-L registers added to the level-L register on which the operation acts.

Since the eight auxiliary level-L registers per level-L register are sufficient for performing each level-L abbreviation (and preparation operation), we attain complete parallelizability; that is, we can perform the level-L abbreviations (and preparation operations) acting on all $\lceil M/K^{(L)} \rceil$ level-L registers in parallel to reduce the time overhead (see Fig. C.1). Note that $R_y(\pm\pi/4)$ gates, preparations, and measurements in a level-L register can be collectively replaced with a single use of the corresponding level-L abbreviation and operations. Similarly, multiple Clifford gates acting on qubits in the same pair of level-L registers can be collectively replaced with a single use of level-L two-register Clifford-gate abbreviation since the level-L Clifford-gate abbreviation can implement an arbitrarily long sequence of Clifford gates in the pair of level-L registers. On the other hand, if a one-depth part of the original circuit includes multiple Clifford gates acting on qubits in different pairs of level-L registers, this part requires multiple level-L two-register Clifford-gate abbreviations, as we will explain in Sec. F.

Then, for each $l = L, L-1, \ldots, 1$, we perform a recursive procedure to compile the level-l circuit into a level-$(l-1)$ circuit, as shown in Fig. C.2. For each level-l register in the level-l circuit, we use, in the corresponding level-$(l-1)$ circuit, a set of N_{r_l} level-$(l-1)$ registers and further add $8+2=10$ auxiliary level-$(l-1)$ registers per set. Similar to the level-L case, among these ten, eight auxiliary level-$(l-1)$ registers are used for the level-$(l-1)$ abbreviations and the level-$(l-1)$ elementary operations in the level-l gadgets. The other two, the level-$(l-1)$ dormant registers, are never used explicitly in the level-$(l-1)$ circuit and are used as the workspace for the level-$(l-1)$ error-correction gadgets. The corresponding level-$(l-1)$ circuit is given by replacing every level-l location of a level-l elementary operation in the level-l circuit with its corresponding level-l gadget and by inserting the level-l error-correction gadgets between all the adjacent pairs of the level-l locations (see Fig. C.2). For deallocated level-l registers, we do not perform error correction; that is, we insert the level-l error-correction gadgets only on the sets of N_{r_l} level-$(l-1)$ registers for the allocated encoded level-l registers. In particular, the dormant level-l registers never require error correction, and we use the corresponding level-$(l-1)$ registers (i.e., $(N_{r_l}+8)$ level-$(l-1)$ registers for each dormant level-l register) as a workspace for the level-l error correction gadgets. The ratio of the allocated level-l registers to the dormant level-l registers is at most $(N_{r_l}+8)/2$ for $l < L$ and is $9/2$ for $l = L$. From (C14), we see that two level-l dormant registers, which corresponds to $2(N_{r_l}+8)$ level-$(l-1)$ usable registers, provide an enough workspace for error correction of an encoded level-l register. The error correction of the encoded level-l registers cannot be done fully in parallel; that is, the level-l dormant registers are time-shared in a series of error-correction gadgets. The length of the series is bounded by $N_{r_l}+8$ for $l < L$ and is 9 for $l = L$.

We perform error correction in a synchronized way as shown in Fig. C.2; that is, we start the level-l error-correction gadgets only after we complete the level-l gadgets for elementary operations over all the allocated encoded level-l registers. During a time period of performing the level-l error-correction gadgets, we do not start performing the other level-l gadgets. In our protocol, the depths of different level-l gadgets may vary. Thus, for the synchronization, we insert wait operations after the level-l gadgets that finish earlier. In particular, let

$$G(l) \tag{C18}$$

be the maximum depth of the level-$(l-1)$ circuit for a level-l gadget, where the maximum is taken over all level-l gadgets including elementary operations and error correction. Then, for each level-l gadget for a level-l elementary operation, the sum of the depth of the gadget itself and that of the wait operations inserted for

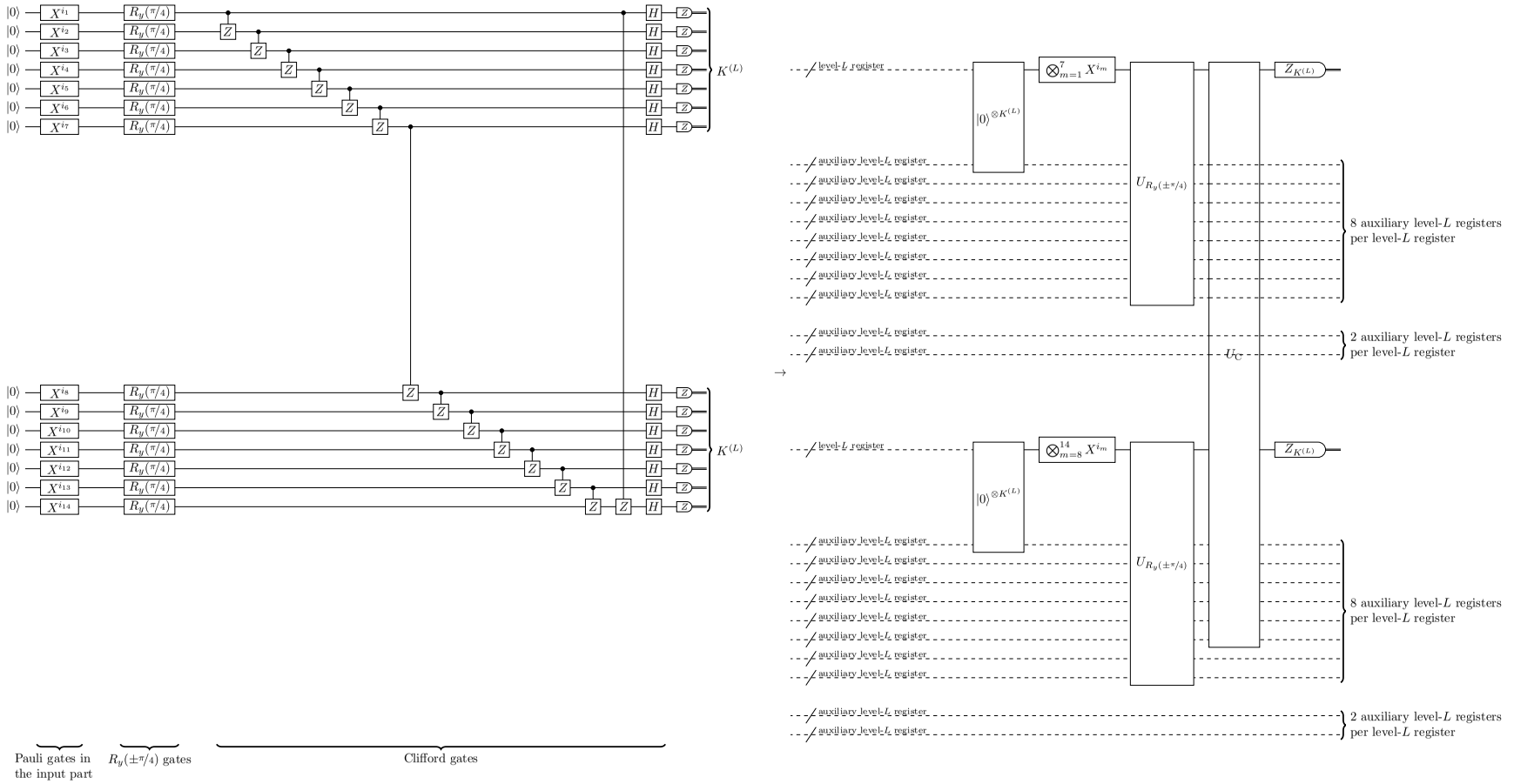


FIG. C.1. An example of 14-qubit original circuits on the left ($W(M) = 14$ with $M = 14$) and the corresponding level-2 circuit obtained from this original circuit on the right ($L = 2$ and $K^{(L)} = 7$). The original circuit is composed of a gate set of Clifford gates X, Y, Z, H, S, CNOT, and CZ, and non-Clifford gates $R_y(\pm\pi/4)$. We represent the $W(M)$ qubits in the original circuit by the qubits in the $\lceil W(M)/K^{(L)} \rceil$ level-L registers and add $8 + 2 = 10$ auxiliary level-L registers per level-L register. Among these ten, eight are used for the level-L abbreviations and the level-L elementary operations as shown in the figure, and the other two are used for error correction as will be explained in Fig. C.2. Pauli gates in the input part (A4) of the original circuit are compiled into level-L Pauli-gate operations. Then, a multiple-depth part of the original circuit composed only of the Clifford gates is compiled into a single use of the level-L two-register Clifford-gate abbreviation $U_{\rm C}$, and a one-depth part of the original circuit with $R_y(\pm\pi/4)$ gates into a single use of the level-L $R_y(\pm\pi/4)$-gate abbreviation $U_{R_y(\pm\pi/4)}$. The abbreviations acting on all the level-L registers can be performed in parallel.

the synchronization is bounded by $G(l)$. As will be shown in Sec. D, we construct gadgets in such a way that $G(l) = O(\mathrm{poly}(N_{r_l}))$. Similarly, wait operations are filled appropriately before and after the error-correction gadgets so that the next level-l gadgets for elementary operations commence synchronously. The depth of the error-correction part of the level-$(l-1)$ circuit is thus upper bounded by $(N_{r_l+1}+8)G(l)$ for $l < L$ and by $9G(l)$ for $l = L$, which are $O(\mathrm{poly}(N_{r_l}))$ in both cases. Note that to optimize the runtime further, non-synchronized time scheduling of error correction could also be considered instead of the synchronized scheduling here while we leave such optimization for future work; for example, it may be possible to perform multiple short-depth level-l gadgets while waiting for another long-depth level-l gadget, but our analysis does not consider such optimization.

By performing this procedure recursively, we obtain the level-0 circuit. Just after performing the above recursive procedure, the level-0 circuit is composed of the level-0 abbreviations and the level-0 elementary operations, which may use auxiliary level-0 registers by definition. However, at level 0, we are allowed to perform operations directly on physical qubits. Thus, we substitute the level-0 two-register Clifford-gate abbreviations in the level-0 circuit with direct applications of the two-qubit Clifford gates and the level-0 $R_y(\pm\pi/4)$-gate abbreviations with the one-qubit $R_y(\pm\pi/4)$ gates, using the operations on physical qubits rather than performing the gate teleportation. After this substitution for the level-0 abbreviations, we also substitute the remaining level-0 elementary operations with the corresponding operations on the physical qubits. As a result, we do not use the $8 + 2 = 10$ auxiliary level-0 registers. Hence after these substitutions, we remove the auxiliary level-0 registers from the level-0 circuit, which yields the fault-tolerant circuit on physical qubits to be executed in our fault-tolerant protocol.

In this compilation, some of the level-l elementary operations and abbreviations are assumed to be invoked with classical arguments, i.e., classical bit strings that dictate the action of the elementary operations and abbreviations. Elementary operations and abbreviations invoked with the classical arguments are called *on-demand* elementary operations and abbreviations. In particular, the Pauli-gate operation (C5) can be invoked with the argument of a classical description of Pauli gates $\bigotimes_{k^{(l)}=1}^{K^{(l)}} P_{k^{(l)}}$, which may not be determined during the compilation but can be given during execution; then, invoked with this classical argument, the Pauli-gate operation can perform the Pauli gates designated by the argument on demand. Also, the Clifford-state preparation operation (C7) and the two-register Clifford-gate abbreviation (C15) can be invoked with the argument of a classical description of the Clifford unitary U_{C} and can then perform, respectively, the corresponding state preparation and the corresponding Clifford gate for U_{C} designated by this argument on demand. The level-l abbreviations pass the classical arguments to the appropriate level-l elementary operations included in the abbreviations, and the level-l gadgets for the level-l elementary operations are designed to be able to translate the arguments from level l to level $(l-1)$ during execution, by performing classical computation. As discussed in Sec. A, the original circuit has the input part (A4), which is implemented by the Pauli gates designated by the input (A1). Also, in the level-l circuit, we may use gate teleportation and error correction, which depend on the outcomes of measurements to be obtained after starting quantum computation. As a whole, on-demand elementary operations and abbreviations are required in the following cases.

1. The level-L Pauli-gate operations used for the input part (A4).
2. The level-l Pauli-gate operations used for correction in the level-l two-register Clifford-gate abbreviation in Fig. D.1 and the level-l error-correction gadget in Fig. D.6.
3. The level-l two-register Clifford-gate abbreviations used for correction in the level-l $R_y(\pm\pi/4)$-gate abbreviation in Fig. D.2.
4. The level-l Clifford-state preparation operations invoked in the level-l on-demand Clifford-gate abbreviations.
5. The level-$(l-1)$ two-register Clifford-state abbreviations invoked in the level-l gadgets of level-l on-demand Clifford-state preparations in Fig. D.7.

It turns out that the above requirement is fulfilled in our protocol if the following set of on-demand elementary operations and abbreviations are available.

1. The level-l on-demand Pauli-gate operations invoked with a $2K^{(l)}$-bit binary row vector $(x_1, z_1, \ldots, x_{K^{(l)}}, z_{K^{(l)}}) \in \{0,1\}^{2K^{(l)}}$ to represent the Pauli gates
$$\bigotimes_{k^{(l)}=1}^{K^{(l)}} P_{k^{(l)}} = \bigotimes_{k^{(l)}=1}^{K^{(l)}} X^{x_{k^{(l)}}} Z^{z_{k^{(l)}}}. \tag{C19}$$
2. The level-l on-demand two-register Clifford-gate abbreviations invoked with $K^{(l)}$ 4×4 binary matrices representing
$$U_{\mathrm{C}} = \bigotimes_{k^{(l)}=1}^{K^{(l)}} U_{\mathrm{C},l}^{(k^{(l)})}, \tag{C20}$$
where $U_{\mathrm{C},l}^{(k^{(l)})}$ is an arbitrary two-qubit Clifford unitary acting on the $k^{(l)}$th qubit in each of the two level-l registers. Here, each $U_{\mathrm{C},l}^{(k^{(l)})}$ is represented as a 4×4 binary matrix in such a way that the conjugation of two-qubit Pauli operators by $U_{\mathrm{C},l}^{(k^{(l)})}$ can be calculated via multiplication of the 4×4 binary

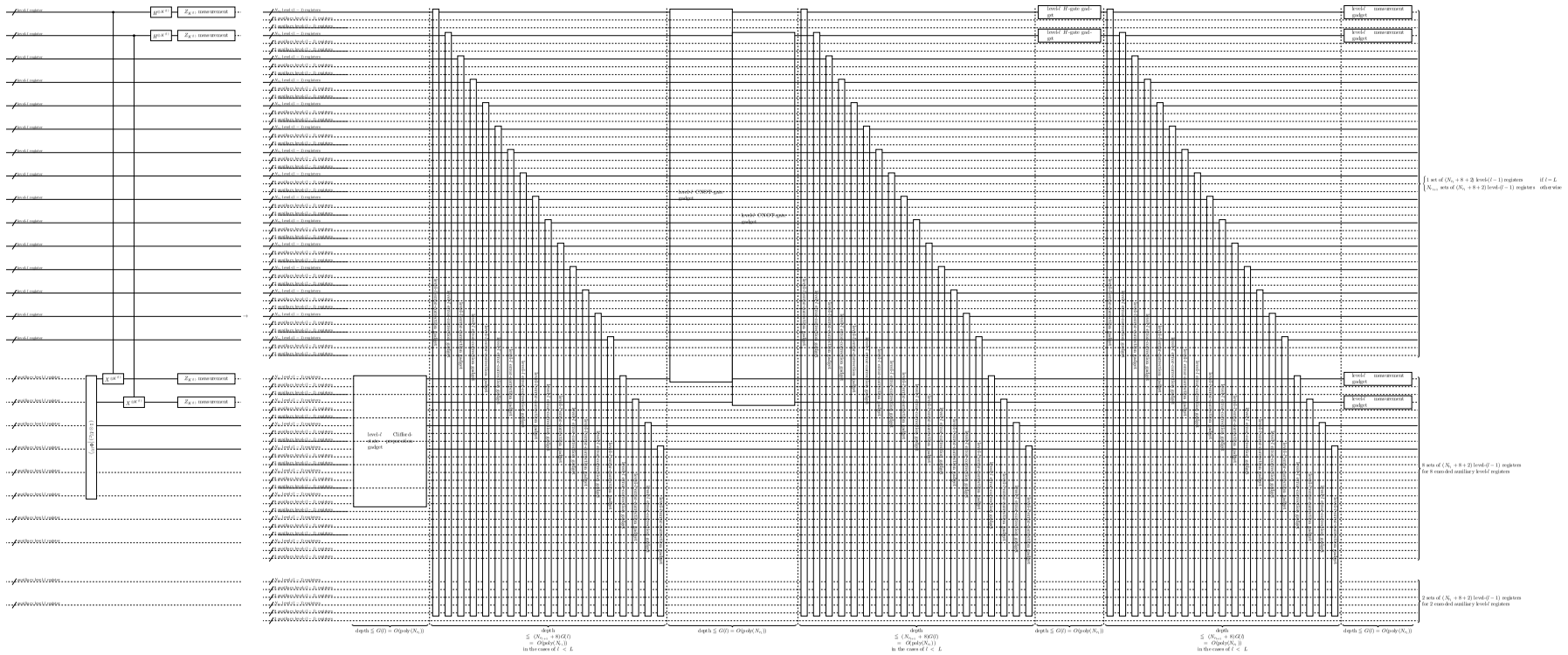


FIG. C.2. The recursive procedure to compile a level-l circuit (left) into a level-$(l-1)$ circuit (right) for $l = L, \ldots, 1$, where we replace each level-l location of the level-l elementary operation with the corresponding level-l gadget and insert the level-l error-correction gadgets in between. The figure shows the case of $l = 1 < L$, i.e., $N_{r_l} = 7$ and $N_{r_{l+1}} = 15$. Note that the figure omits the double-line wires of measurement operations (C1) for representing the measurement outcome. The level-L circuit is given by the procedure of Fig. C.1. Then for $l = L, \ldots, 1$, in place of each level-l register in the level-l circuit (e.g., each of the solid and dashed wires on the right of Fig. C.1), we use a set of N_{r_l} level-$(l-1)$ registers and add $8 + 2 = 10$ auxiliary level-$(l-1)$ registers per set in the corresponding level-$(l-1)$ circuit. Among these ten auxiliary level-$(l-1)$ registers, eight are used for level-$(l-1)$ elementary operations and level-$(l-1)$ abbreviations, and the other two are level-$(l-1)$ dormant registers that are never used explicitly in the level-$(l-1)$ circuit but will be used for level-$(l-1)$ error-correction gadgets. The level-l elementary operations are replaced with the corresponding level-l gadgets, which are level-$(l-1)$ circuits to perform the elementary operation on the encoded level-l registers. Then, level-l error-correction gadgets are inserted in between for all the allocated encoded level-l registers in a synchronized way; that is, during each time period of performing the level-l error-correction gadgets on all the encoded level-l registers, we do not perform the other level-l gadgets, as shown in the figure. For the synchronization, we may insert wait operations before and after each level-l gadget to wait for the timing of the error correction. The maximal depth among all the level-l gadgets is denoted by $G(l)$, which satisfies $G(l) = O(\mathrm{poly}(N_{r_l}))$ in our protocol. Then, the sum of the depth of each gadget and the wait operations is upper bounded by $G(l)$. In each time period of the error correction, we perform the error-correction gadgets sequentially by reusing the workspace of the two auxiliary encoded level-l registers. In the case of $l = L$, for allocated registers among each encoded level-L register and its eight auxiliary level-L registers for elementary operations, we perform (at most) $1 + 8 = 9$ level-L error-correction gadgets sequentially. The depth of each part of the error correction in the level-$(L-1)$ circuit is bounded by $9G(l) = O(\mathrm{poly}(N_{r_l}))$. In the cases of $l < L$, for allocated registers among each set of $N_{r_{l+1}}$ encoded level-l registers and their eight auxiliary level-l registers for elementary operations, we perform (at most) $N_{r_{l+1}} + 8$ level-l error-correction gadgets sequentially. In these cases, the depth of each part of the error correction in the level-$(l-1)$ circuit is bounded by $(N_{r_{l+1}} + 8)G(l) = O(\mathrm{poly}(N_{r_l}))$.

matrix representing $U_{\mathrm{C},l}^{(k^{(l)})}$ (from the right of the 4-bit binary row vector representing the two-qubit Pauli operators), as shown in Ref. [32].

3. The level-l on-demand Clifford-state preparations invoked with the $K^{(l)}$ 4×4 binary matrices representing U_{C} in the form of (C20).

We will construct abbreviations and gadgets in Sec. D in such a way that the abbreviations and gadgets offer the above on-demand functions and are implemented by using only the above on-demand functions. Prior to starting the execution of quantum computation, the compilation represents these parts of the fault-tolerant circuit in terms of on-demand elementary operations to be invoked with the classical arguments, and all the other parts of the circuits are given by fixed circuits without such classical arguments, referred to as fixed parts. The fixed parts may also include the Pauli-gate operation, the Clifford-state preparation operations, and the two-register Clifford-gate abbreviations that are not invoked with the classical arguments. In the fixed parts, the actions of all the elementary operations and the abbreviations are determined during the compilation so as to reduce the time overhead of waiting for the classical computation during execution.

Appendix D: Fault-tolerant gadgets and abbreviations

In this section, after defining the conditions of fault-tolerant gadgets, the detail of the construction of the gadgets and the abbreviations in our fault-tolerant protocol is explained one by one, using Figs. D.1-D.9 to show the correspondence. To show an abbreviation, the corresponding level-l circuit in terms of level-l elementary operations will be given in the figures. In particular, the two-register Clifford-gate abbreviation is given by Fig. D.1, and the $R_y(\pm\pi/4)$-gate abbreviation by Fig. D.2. As for a gadget, the corresponding level-$(l-1)$ circuit of the gadget will be given in the figures. In particular, the measurement gadget is given by Fig. D.3, the H-, CNOT-, CZ-, and Pauli-gate gadgets by Fig. D.4, the initial-state preparation gadget by Fig. D.5, the error-correction gadget by Fig. D.6, the Clifford-state preparation gadget by Fig D.7, and the magic-state preparation gadget by Fig. D.9. For explicitness, level-$(l-1)$ circuits of level-l gadgets are shown using $N_{r_l}=7$ and $K_{r_l}=1$ as in the case of $l=1$. To simplify the description of the level-$(l-1)$ circuits, gadgets may refer to other gadgets as sub-gadgets; in such a case, the level-$(l-1)$ circuit may include a box with the name of another level-l gadget, which is to be replaced with the level-$(l-1)$ circuit of the level-l gadget of the box. In addition, the level-$(l-1)$ circuit may include level-$(l-1)$ abbreviations, which should be replaced with the corresponding level-$(l-1)$ circuits. Similar to sub-gadgets, gadgets may refer to a combination of other sub-gadgets as a sub-abbreviation; in such a case, the level-$(l-1)$ circuit may include a box with the name of a level-l abbreviation, which should be considered to be replaced with the level-$(l-1)$ circuit obtained from the level-l circuit of the level-l abbreviation of the box by replacing each level-l elementary operation with the corresponding level-l gadget.

Conditions of fault-tolerant gadgets: We define the properties of gadgets required for fault tolerance. To define such requirements, as in Ref. [1], we introduce an ideal decoder and r-filters for our protocol. In Ref. [1], the conditions for fault-tolerant gadgets are given in the form of equivalence relations between circuits including the gadget, the r-filters, the ideal decoder, and the ideal intended elementary operation, conditioned on the number of faulty locations in the gadget. Although Ref. [1] provides concrete definitions for the ideal decoder and the r-filters in the case of a $[[N,1,2t+1]]$ code for $0\leqq r\leqq t$, the proof of the threshold theorem does not rely on specific definitions but on the equivalence relations alone. Thus, to use the same argument as the proof of the threshold theorem in Ref. [1], we will give appropriately modified definitions of an ideal filter and r-filters so that the gadgets proposed here should satisfy effectively the same set of equivalence relations as those in Ref. [1].

An ideal decoder, a 0-filter, and a 1-filter in our case ($t=1$) are defined as follows. As explained in Sec. B, a level-l register is encoded into $N_{r_l}K^{(l-1)}$ qubits in N_{r_l} level-$(l-1)$ registers, and the $k^{(l-1)}$th qubit of the nth level-$(l-1)$ register is labeled $(n,k^{(l-1)})$ with $k^{(l-1)}\in\{1,\dots,K^{(l-1)}\}$ and $n\in\{1,\dots,N_{r_l}\}$. Let

$$\mathcal{H}_{n,k^{(l-1)}}\cong\mathbb{C}^2 \tag{D1}$$

be the Hilbert space of the qubit $(n,k^{(l-1)})$ in a level-$(l-1)$ register. Define

$$\mathcal{H}_{k^{(l-1)}}:=\bigotimes_{n=1}^{N_{r_l}}\mathcal{H}_{n,k^{(l-1)}}, \tag{D2}$$

$$\mathcal{H}:=\bigotimes_{k^{(l-1)}=1}^{K^{(l-1)}}\mathcal{H}_{k^{(l-1)}}, \tag{D3}$$

where $\mathcal{H}$ is the whole space of the N_{r_l} level-$(l-1)$ registers. The quantum Hamming code $\mathcal{Q}_{r_l}$ defines a code space for each $k^{(l-1)}$

$$\mathcal{H}^{\mathrm{code}}_{k^{(l-1)}}\subset\mathcal{H}_{k^{(l-1)}}. \tag{D4}$$

Then the whole code space in the N_{r_l} level-$(l-1)$ registers is given by

$$\mathcal{H}^{\mathrm{code}}:=\bigotimes_{k^{(l-1)}=1}^{K^{(l-1)}}\mathcal{H}^{\mathrm{code}}_{k^{(l-1)}}\subset\mathcal{H}. \tag{D5}$$

The level-l 0-filter acting on the N_{r_l} level-$(l-1)$ registers is defined as a projector

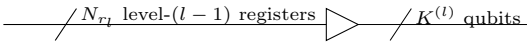
$$\Pi^{\mathrm{code}}\text{ onto the subspace }\mathcal{H}^{\mathrm{code}}\subset\mathcal{H}. \tag{D6}$$

Suppose that a codeword $|\psi_{k^{(l-1)}}\rangle \in \mathcal{H}^{\rm code}_{k^{(l-1)}}$ suffers from an error represented by a Pauli operator $P_{k^{(l-1)}} \in \mathcal{P}_{N_{r_l}}$ acting on $\mathcal{H}_{k^{(l-1)}}$. If the weight of $P_{k^{(l-1)}}$ is at most 1, the code $\mathcal{Q}_{r_l}$ can correct the error. As a result, if a codeword $|\psi\rangle \in \mathcal{H}^{\rm code}$ suffers from an error represented by $P = \bigotimes_{k^{(l-1)}=1}^{K^{(l-1)}} P_{k^{(l-1)}}$, the $K^{(l-1)}$ blocks of the code $\mathcal{Q}_{r_l}$ can correct the error P as long as every $P_{k^{(l-1)}}$ has a weight at most 1. Such an error P is said to be *correctable* here. An ideal decoder here is defined to do this correction and decoding; that is, the ideal decoder receives the N_{r_l} level-$(l-1)$ registers in an input state $P|\psi\rangle$, corrects errors to recover the state $|\psi\rangle$, and converts it to the non-encoded state of a level-l register. The level-l 1-filter acting on the N_{r_l} level-$(l-1)$ registers is defined as a projector onto the subspace spanned by all the states in the form $P|\psi\rangle$ with P correctable and $|\psi\rangle \in \mathcal{H}^{\rm code}$. It turns out that this subspace is the entire space $\mathcal{H}$, and the 1-filter is trivial, i.e.,

$$\mathbb{1}^{\otimes N_{r_l} K^{(l-1)}} \text{ on } \mathcal{H}. \tag{D7}$$

With these definitions of the ideal decoder and the r-filters ($r = 0, 1$), the notations used for the condition for a gadget to be fault-tolerant are introduced as follows. Note that the notations here are made analogous to Ref. [1] using qubits, but are different from Ref. [1] in that each wire represents one or multiple registers. The ideal decoder is denoted by

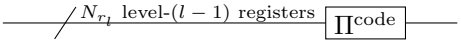
ideal decoder:



, (D8)

where the thin wire after the ideal decoder represents a level-l register composed of $K^{(l)} = K_{r_l} K^{(l-1)}$ qubits, and each of the other wires with regular thickness represents N_{r_l} level-$(l-1)$ registers collectively. The 0-filter here reduces to $\Pi^{\rm code}$ as shown in (D6), and is denoted by

0-filter:



. (D9)

The 1-filter is not illustrated explicitly since it is trivial as shown in (D7). The variable s placed in the upper right of a gadget represents the number of faulty locations in the gadget. An operation depicted using thin lines and acting on thin wires is an ideal operation. With these notations, every nontrivial condition for a gadget to be fault-tolerant given in Ref. [1] is translated as follows, where dashed wires are omitted for simplicity of presentation.

The measurement gadget is fault-tolerant if it satisfies

meas: when $s = 0$

,

and when $s = 0, 1$

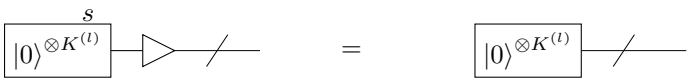
. (D10)

The initial-state preparation gadget is fault-tolerant if it satisfies

prep A: when $s = 0$

, (D11)

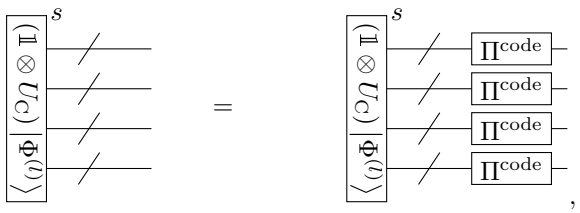
prep B: when $s = 0, 1$



. (D12)

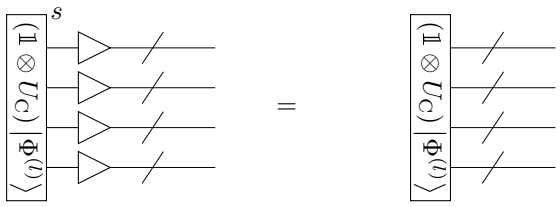
The Clifford-state preparation gadget is fault-tolerant if it satisfies

prep A: when $s = 0$



, (D13)

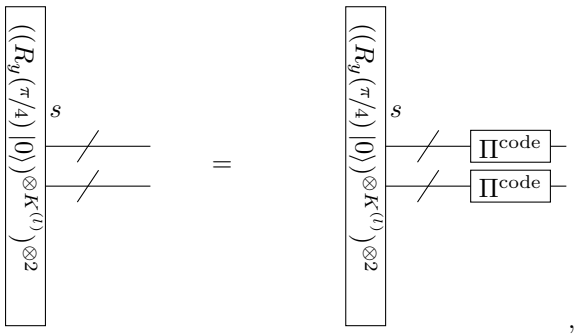
prep B: when $s = 0, 1$



. (D14)

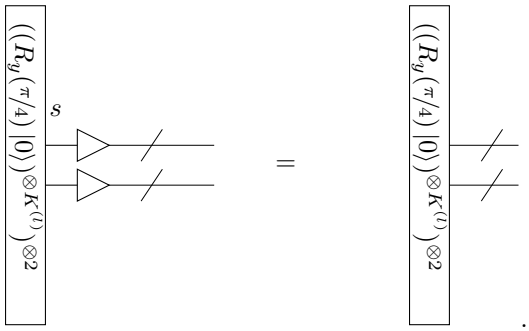
The magic-state preparation gadget is fault-tolerant if it satisfies

prep A: when $s = 0$



, (D15)

prep B: when $s = 0, 1$



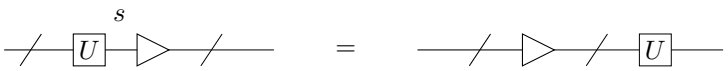
. (D16)

The H- or Pauli-gate gadget is fault-tolerant if it satisfies

gate A: when $s = 0$

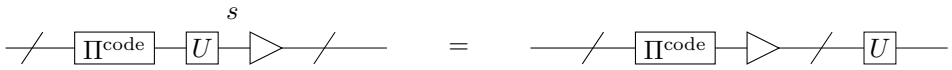
, (D17)

gate B: when $s = 0$



,

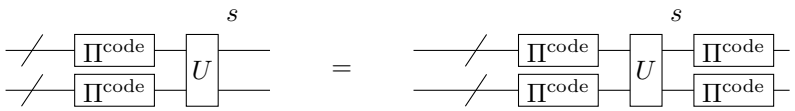
and when $s = 0, 1$



, (D18)

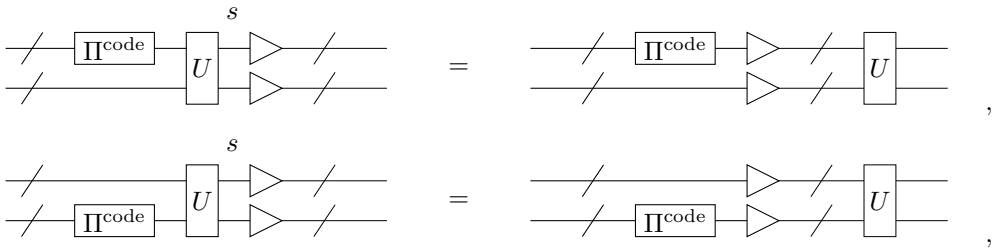
where U is the logical gate applied by the gadget. The CNOT- or CZ-gate gadget is fault-tolerant if it satisfies

gate A: when $s = 0$

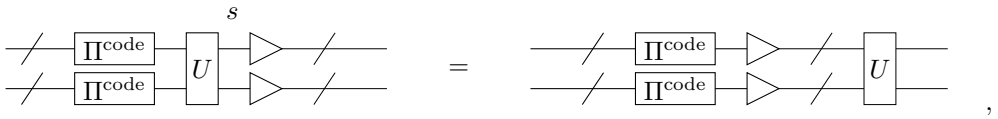


, (D19)

gate B: when $s = 0$



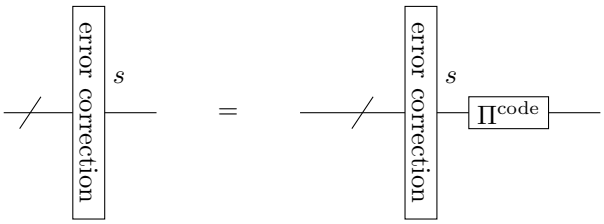
and when $s = 0, 1$



, (D20)

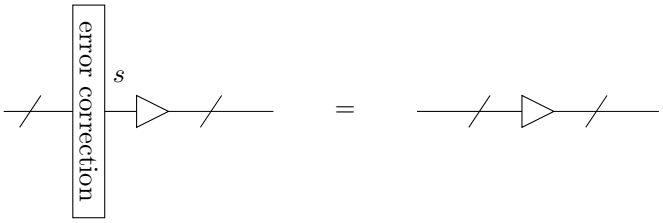
where U is the logical gate applied by the gadget. An error-correction gadget is fault-tolerant if it satisfies

EC A: when $s = 0$



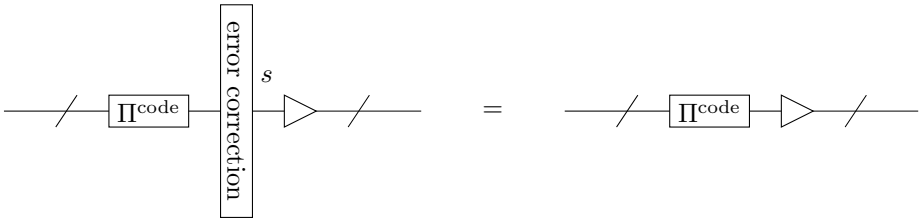
, (D21)

EC B: when $s = 0$



,

and when $s = 0, 1$



. (D22)

These conditions of fault tolerance are used in place of the conditions of fault tolerance in Ref. [1], where the labels of the conditions here in the bold text show the corresponding conditions in Ref. [1] with the same labels.

Given that every gadget is fault-tolerant, the existence of a threshold can be proved for the local statistic error

model [1]. As will be explained in Sec. E, it is possible to apply the same argument as that in Ref. [1] to prove that the level-l circuit at every level $l = 0, 1, \ldots, L$ follows the local stochastic error model with a bound $p_l > 0$ on the logical error rate, i.e., the probability of having a fault at a given level-l location. To determine the scaling of p_l in Sec. E, it is necessary to have a bound on $G(l)$ defined in (C18), i.e., the maximum depth of the level-$(l-1)$ circuit for a level-l gadget, where the maximum is taken over all level-l gadgets including elementary operations and error correction.

In the explanation of each gadget below, we will check that

$$G(l) = O(\mathrm{poly}(N_{r_l})). \tag{D23}$$

From (B19), it then follows that

$$G(l) = \exp(O(l)). \tag{D24}$$

The depth of each gadget may include wait operations to wait for classical computation such as those for the decoder and the gate teleportation. During our protocol, the number of parallel processes for classical computation is limited to

$$O(\mathrm{poly}(N^{(l)})), \tag{D25}$$

which will turn out to satisfy the condition (A8) on the number of parallel processes, as will be shown in Sec. F. We will also show that the depth of the level-l circuit for a level-l abbreviation is upper bounded by

$$O(\log(N^{(l)})), \tag{D26}$$

including the wait operations to wait for the nonzero-time classical computation in the gate teleportation. To obtain these bounds, our analysis in the following uses

$$K^{(l)} \approx N^{(l)}, \tag{D27}$$
$$K_{r_l} \approx N_{r_l}, \tag{D28}$$

due to (B35) and (B6), and

$$r_l \approx \log(N_{r_l}) = O(l) \tag{D29}$$
$$\ll \log(N^{(l)}) = O(l^2) \tag{D30}$$
$$\ll \mathrm{poly}(N_{r_l}) = \exp(O(l)) \tag{D31}$$
$$\ll \mathrm{poly}(N^{(l)}) = \exp(O(l^2)) \tag{D32}$$

due to (B19) and (B27).

To bound the space overhead, it is also necessary to have bounds on the number of registers in each level-l gadget. As designed in Sec. C, in our protocol, the number of level-$(l-1)$ registers used in a level-l gadget is to be bounded by

$$N_{r_l} + 8 + 2 = N_{r_l} + O(1) \tag{D33}$$

per encoded level-l register. The number of bits used for each level-l gadget is to be bounded by

$$O(\mathrm{poly}(N^{(l)})). \tag{D34}$$

Two-register Clifford-gate abbreviation: The level-l two-register Clifford-gate abbreviation based on gate teleportation [33–35] is given by Fig. D.1, where the gate teleportation is also presented. The implementation is assisted by auxiliary level-l registers in a state $(\mathbb{1}^{B_1B_2} \otimes U_{\mathrm{C}}^{B_3B_4}) |\Phi^{(l)}\rangle^{B_1B_2B_3B_4}$, which is prepared by the Clifford-state preparation operation in (C7). Given the $4K^{(l)}$-bit measurement outcome, correction in quantum teleportation [36] is given by an operator in the form of

$$U_{\mathrm{C}} \left(\bigotimes_{k^{(l)}=1}^{K^{(l)}} P_{k^{(l)}}^{B_3} \otimes \bigotimes_{k^{(l)}=1}^{K^{(l)}} P_{k^{(l)}}^{B_4} \right) U_{\mathrm{C}}^{\dagger}, \tag{D35}$$

where $P_{k^{(l)}}^{B_j} \in \{X, Z, Y, \mathbb{1}\}$ for each $j \in \{3, 4\}$ and $k^{(l-1)} \in \{1, \ldots, K^{(l-1)}\}$ is a Pauli gate acting on the $k^{(l)}$th qubit of the level-l register B_j with labeling (B23). Since U_{C} is Clifford, the operator in (D35) is a tensor product of single-qubit Pauli operators. Conditioned on the measurement outcome, the tensor product of Pauli operators (D35) acting on the level-l registers is calculated using an efficient decoder and applied as the correction of the quantum teleportation by the Pauli-gate operation in (C5) [34, 35]. The $4K^{(l)}$-bit measurement outcome yields a $4K^{(l)}$-dimensional binary row vector representing the $2K^{(l)}$-qubit Pauli operator $\bigotimes_{k^{(l)}=1}^{K^{(l)}} P_{k^{(l)}}^{B_3} \otimes \bigotimes_{k^{(l)}=1}^{K^{(l)}} P_{k^{(l)}}^{B_4}$ in (D35), and we can compute its conjugation (D35) by the Clifford unitary U_{C} via multiplication of a $4K^{(l)} \times 4K^{(l)}$ binary matrix representing U_{C} (from the right of the row vector), as shown in Ref. [32]. The resulting $4K^{(l)}$-dimensional binary row vector of (D35) is used as the classical argument in invoking the Pauli-gate operation for the correction.

The level-l circuit of the abbreviation has the depth bounded by

$$O(\log(N^{(l)})). \tag{D36}$$

The depth is dominated by the wait operations to wait for classical computation to obtain the tensor product of Pauli operators (D35) since the other part has a constant depth. Regarding the runtime of classical computation, using $O(N^{(l)2})$ parallel processes, we can perform the multiplication of the $4K^{(l)} \times 4K^{(l)}$ binary matrix for the conjugation (D35) within runtime $O(\log(K^{(l)})) = O(\log(N^{(l)}))$, which leads to (D36). The required number of bits is $O(K^{(l)2}) = O(N^{(l)2})$, which is dominated by those for storing the $4K^{(l)} \times 4K^{(l)}$ matrix representing U_{C}.

As shown in (C20), the level-l two-register Clifford-gate abbreviation can be invoked with the argument of the classical description of U_{C}, and we here describe its on-demand function. In the on-demand case, the argument is passed to the level-l Clifford-state preparation operation used in the abbreviation. Since we represent U_{C} in (C20) as $K^{(l)}$ 4×4 binary matrices, we can compute the conjugation (D35) within $O(1)$ runtime, using $O(N^{(l)})$ parallel processes of multiplying

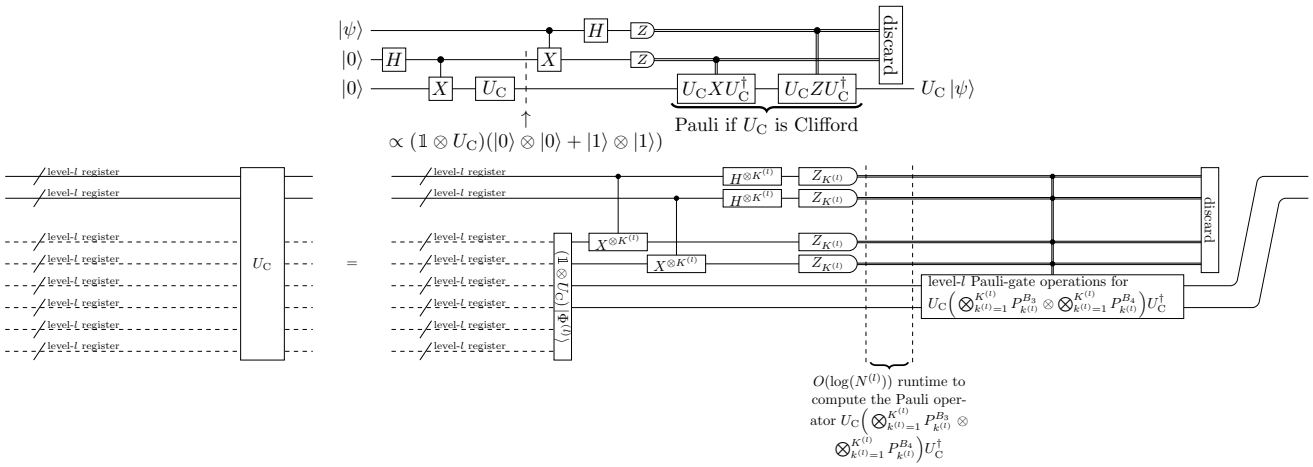


FIG. D.1. Gate teleportation for implementing a single-qubit Clifford gate at the top, and the level-l two-register Clifford-gate abbreviation based on the gate teleportation for applying a Clifford gate U_C on two level-l registers at the bottom. The implementation is assisted by auxiliary level-l registers in a state $(\mathbb{1}^{B_1B_2}\otimes U_C^{B_3B_4})\,|\Phi^{(l)}\rangle^{B_1B_2B_3B_4}$, which is prepared by the Clifford-state preparation operation in (C7). In the figure, the level-l registers B_1, B_2, B_3, B_4 are shown from top to bottom. Conditioned on the measurement outcome, a tensor product of Pauli operators $U_C\left(\bigotimes_{k^{(l)}=1}^{K^{(l)}} P_{k^{(l)}}^{B_3}\otimes\bigotimes_{k^{(l)}=1}^{K^{(l)}} P_{k^{(l)}}^{B_4}\right)U_C^\dagger$ in (D35) is calculated and applied as the correction in the quantum teleportation by two level-l Pauli-gate operations in (C5). If invoked with the argument dictating U_C, the abbreviation passes the argument to the Clifford-state preparation operation in the circuit.

these $K^{(l)} = O(N^{(l)})$ constant-size binary matrices in parallel. Therefore, the part of the level-l circuit relevant to processing the argument has the depth $O(1)$, using $O(K^{(l)}) = O(N^{(l)})$ bits. As a whole, (D36) is an upper bound of the depth even in the on-demand case.

$R_y(\pm\pi/4)$-gate abbreviation: The level-l $R_y(\pm\pi/4)$-gate abbreviation based on gate teleportation is given by Fig. D.2, where the gate teleportation is also presented. The implementation is assisted by two types of auxiliary level-l registers A_1 and A_2. The register A_1 works like the auxiliary qubit in the gate teleportation of Fig. D.2 and is initially prepared in a magic state $(R_y(\pi/4)\,|0\rangle)^{\otimes K^{(l)}}$ by the magic-state preparation operation in (C8). Note that, since the magic-state preparation operation prepares two registers in $((R_y(\pi/4)\,|0\rangle)^{\otimes K^{(l)}})^{\otimes 2}$, the circuit in Fig. D.2 discards one of the prepared magic states for simplicity of presentation; however, we may use the magic state discarded in Fig. D.2 as that of A_1 for another $R_y(\pm\pi/4)$-gate abbreviation to improve the efficiency. The other register A_2 is used for applying the $R_y(\pi/4)$ gates only to the selected qubits and is initially prepared in a state $|0\rangle^{\otimes K^{(l)}}$ by the initial-state preparation operation in (C6). For each qubit on which $U_{R_y(\pm\pi/4)}$ acts trivially as $\mathbb{1}$, a single-qubit SWAP gate is applied between A_1 and A_2, so as to avoid the subsequent application of the $R_y(\pi/4)$ gate. These SWAP gates are implemented collectively by a single use of a two-register Clifford-gate abbreviation in Fig. D.1. For example, to implement $U_{R_y(\pm\pi/4)} = (R_y(\pm\pi/4))^{\otimes K^{(l)}}$, no SWAP gate is applied, and to implement $U_{R_y(\pm\pi/4)} = R_y(\pm\pi/4)^{\otimes(K^{(l)}-1)}\otimes\mathbb{1}$, a SWAP gate is applied between a pair of the qubits. Then the gate teleportation to apply $R_y(\pi/4)$ is performed by the circuit at the top of Fig D.2. Conditioned on the measurement outcome, Clifford gates for the correction, i.e., HZ gates, are applied. At this point, to the qubits on which $U_{R_y(\pm\pi/4)}$ dictates to apply $R_y(-\pi/4)$, we have applied $R_y(\pi/4)$ gates. Thus, an extra Clifford gate $R_y(-\pi/2) = ZH$ is applied to each of these qubits in the end. Each of the corrections of HZ gates and the application of extra $R_y(-\pi/2)$ gates are implemented by a two-register Clifford-gate abbreviation in Fig. D.1. These gates may act nontrivially only on a single register, but for each $k^{(l)} \in \{1,\dots,K^{(l)}\}$, we here represent a HZ and $R_y(-\pi/2)$ (or $\mathbb{1}$) gate for the correction acting on the $k^{(l)}$th qubit in the level-l register as a two-qubit Clifford unitary $HZ\otimes\mathbb{1}$ and $R_y(-\pi/2)\otimes\mathbb{1}$ (or $\mathbb{1}\otimes\mathbb{1}$) acting trivially on the $k^{(l)}$th qubit of another level-l register as well (i.e., A_2 in Fig. D.2). The correction of HZ gates is applied by the on-demand two-register Clifford gate abbreviation with (C20).

The level-l circuit for the implementation has the depth

$$O(\log(N^{(l)})). \tag{D37}$$

The depth is dominated by the two-register Clifford-gate abbreviations since the other part has a constant depth. Using the $K^{(l)}$-bit outcome of the measurement, we perform the classical computation to decide the correction by HZ gates as shown in Fig. D.2. Performing this classical computation for all $k^{(l)}$, we obtain the $K^{(l)}$ 4×4 matrices as in (C20), which we use as the argument to invoke the two-register Clifford-gate abbreviation for performing the correction. Using $K^{(l)} = O(N^{(l)})$ parallel processes, the runtime of this computation is $O(1)$,

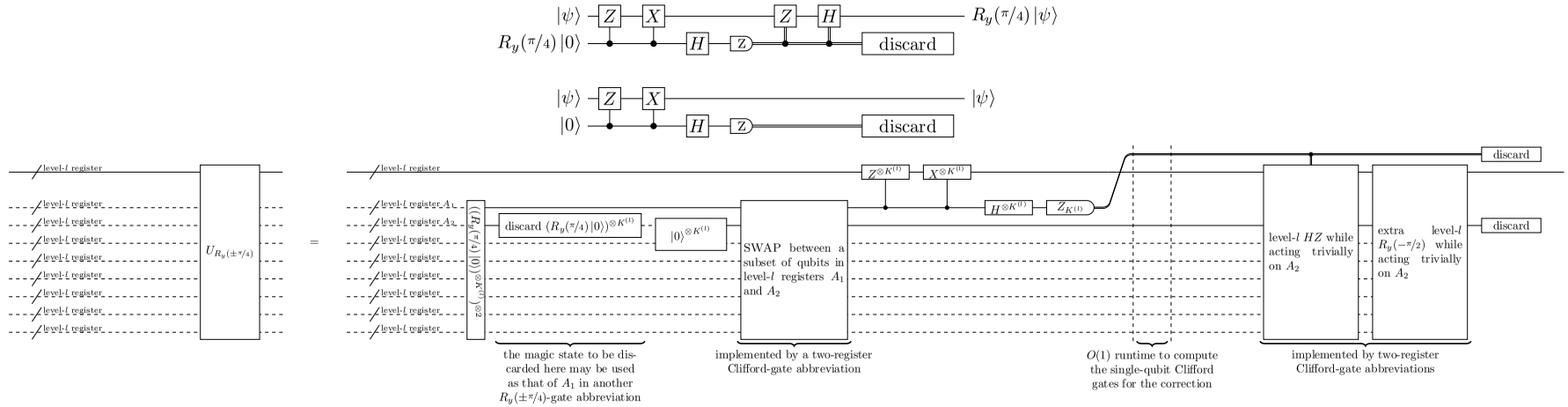


FIG. D.2. Gate teleportation assisted by a magic state $R_y(\pi/4)|0\rangle$ for performing $R_y(\pi/4)$ at the top, the same circuit as the gate teleportation assisted by $|0\rangle$ for acting trivially as $\mathbb{1}$ in the middle, and the level-l $R_y(\pm\pi/4)$-gate abbreviation based on the gate teleportation for applying $U_{R_y(\pm\pi/4)}$ gates to a level-l register at the bottom, where $U_{R_y(\pm\pi/4)}$ is an arbitrary tensor product of $R_y(\pi/4)$, $R_y(-\pi/4)$, and $\mathbb{1}$. The implementation is assisted by two types of auxiliary level-l registers A_1 and A_2. The registers A_1 and A_2 are prepared in $(R_y(\pi/4)|0\rangle)^{\otimes K^{(l)}}$ and $|0\rangle^{\otimes K^{(l)}}$ by the magic-state preparation operation in (C8) and by the initial-state preparation operation in (C6), respectively. For each qubit on which $U_{R_y(\pm\pi/4)}$ acts trivially as $\mathbb{1}$, a SWAP gate is applied between A_1 and A_2, so as to avoid the subsequent application of the $R_y(\pm\pi/4)$ gates. These SWAP gates are collectively implemented by a two-register Clifford-gate abbreviation in Fig. D.1. Then, the gate teleportation for applying $R_y(\pi/4)$ and $\mathbb{1}$ is performed. Conditioned on the measurement outcome, level-l Clifford gates for the correction, i.e., HZ gates, are applied. To perform $R_y(-\pi/4)$, we use the circuit at the top with an extra Clifford gate $R_y(-\pi/2) = ZH$ applied in the end after the correction with the HZ gate. Each of the correction HZ and the extra $R_y(-\pi/2)$ is implemented by a two-register Clifford-gate abbreviation in Fig. D.1 acting trivially on A_2 that is to be discarded in the end.

where, from each of the $K^{(l)}$ bits of the measurement outcome, one of the $K^{(l)}$ processes computes each 4×4 (i.e., constant-size) matrix. Apart from the classical computation, the depth of each two-register Clifford-gate abbreviation is upper bounded by $O(\log(N^{(l)}))$ as shown in (D36), which is dominant. The required number of bits is $O(K^{(l)}) = O(N^{(l)})$, which is dominated by those for storing the $K^{(l)}$ 4×4 matrices used for the two-register Clifford gates.

Measurement gadget: The level-l measurement gadget is constructed as in Fig. D.3. The gadget is implemented by transversally performing level-$(l-1)$ measurement operations, followed by an efficient decoder. The fault tolerance, i.e., (D10), follows from the transversality.

We here provide the construction of the efficient decoder for the quantum Hamming codes to analyze its runtime. In particular, we provide a protocol for calculating the measurement outcomes for $K^{(l)}$ qubits in a level-l register from those in the N_{r_l} level-$(l-1)$ registers, where the outcomes at the physical level ($l-1=0$) are those of the measurement in the Z basis of the physical qubits. Let $o_{n,k^{(l-1)}} \in \{0,1\}$ be the measurement outcome of the qubit $(n,k^{(l-1)})$ with (B24) in the level-$(l-1)$ registers ($n\in\{1,\dots,N_{r_l}\}$ and $k^{(l-1)}\in\{1,\dots,K^{(l-1)}\}$), which is determined by the level-$(l-1)$ measurement operations. For each $k^{(l-1)}$, we estimate the true outcome by the following procedure. We calculate the syndrome $e_{i,k^{(l-1)}}\in\{0,1\}$ for each $i\in\{1,\dots,r_l\}$ by

$$e_{i,k^{(l-1)}} = \sum_{n=1}^{N_{r_l}} o_{n,k^{(l-1)}} b_{i,n} \mod 2, \tag{D38}$$

where $b_{i,n}$ is the element of the parity-check matrix used in (B2) for $\mathcal{Q}_{r_l}$. This calculation provides the label of the presumably erroneous qubit written as

$$\tilde{n}_{k^{(l-1)}} \coloneqq \sum_{i=1}^{r_l} e_{i,k^{(l-1)}} 2^{i-1}, \tag{D39}$$

which is the bit string stored in the memory after the calculation of (D38). If $\tilde{n}_{k^{(l-1)}}=0$, then we estimate that $o_{1,k^{(l-1)}},\dots,o_{N_{r_l},k^{(l-1)}}$ have no error; otherwise, we estimate that the qubit $(\tilde{n}_{k^{(l-1)}},k^{(l-1)})$ has an error. To calculate the measurement outcome for the level-l register, we use the labeling (B23), i.e., $k^{(l)}\mapsto(k,k^{(l-1)})$, for the $k^{(l)}$th qubit in the level-l register, where $k^{(l)}\in\{1,\dots,K^{(l)}\}$ and $k\in\{1,\dots,K_{r_l}\}$.

Under this labeling $k^{(l)}\mapsto(k,k^{(l-1)})$, for each $k^{(l)}$, we calculate the measurement outcome $o_{k^{(l)}}$ of the $k^{(l)}$th qubit in the level-l register from those of $(n,k^{(l-1)})$ in the level-$(l-1)$ registers as

$$o_{k^{(l)}} = \begin{cases} \sum_{n=1}^{N_{r_l}} o_{n,k^{(l-1)}} b_n^{(k)} & \mod 2,\ \text{if } \tilde{n}_{k^{(l-1)}}=0,\\ \sum_{n=1}^{N_{r_l}} o_{n,k^{(l-1)}} b_n^{(k)} + b^{(k)}_{\tilde{n}_{k^{(l-1)}}} & \mod 2,\ \text{otherwise,}\end{cases} \tag{D40}$$

where $b_n^{(k)}$ is the element of the bit string in (B7) for representing the kth logical qubit of $\mathcal{Q}_{r_l}$. Then, the decoder outputs the level-l measurement outcomes $o_{k^{(l)}}$ while all the level-$(l-1)$ measurement outcomes $o_{n,k^{(l-1)}}$ are deallocated.

The gadget has a constant depth

$$O(1), \tag{D41}$$

which is followed by classical computation for the decoder to calculate the measurement outcome for the encoded level-l register. We also bound the runtime of this classical computation in the following. To calculate the measurement outcomes with the error correction, we use $r_l K^{(l-1)}\times N_{r_l}$ processes of classical computation in parallel to obtain the syndromes $e_{i,k^{(l-1)}}$ in (D38) for all $i\in\{1,\dots,r_l\}$ and $k^{(l-1)}\in\{1,\dots,K^{(l-1)}\}$. We divide these $r_lK^{(l-1)}\times N_{r_l}$ processes into $r_lK^{(l-1)}$ sets of N_{r_l} processes. Using each set, we calculate $e_{i,k^{(l-1)}}$ by parallel classical computation with the N_{r_l} processes in the set. The runtime of the sum in (D38) for this calculation is $O(\log(N_{r_l}))$. Therefore, using at most $r_lK^{(l-1)}\times N_{r_l} = O(N^{(l)}\log(N_{r_l}))$ processes (where we use $K^{(l-1)}N_{r_l} = O(N^{(l-1)}N_{r_l} = N^{(l)})$), we can obtain $\tilde{n}_{k^{(l-1)}}$ in (D39) in the memory within runtime $O(\log(N_{r_l}))$. In addition, we use $K^{(l)}\times N_{r_l}$ processes of classical computation in parallel to obtain the measurement outcomes $o_{k^{(l)}}$ in (D40) for all $k^{(l)}$. We divide these $K^{(l)}\times N_{r_l}$ processes into $K^{(l)}$ sets of N_{r_l} processes. Using each set, we calculate the sum over n and the error correction of $b^{(k)}_{\tilde{n}_{k^{(l-1)}}}$ in (D40) by parallel classical computation with the N_{r_l} processes in the set, which can be performed within the runtime of $O(\log(N_{r_l}))$. As a whole, using at most $K^{(l)}\times N_{r_l} = O(N^{(l)}N_{r_l})$ processes, we can obtain $o_{k^{(l)}}$ within runtime

$$O(\log(N_{r_l})). \tag{D42}$$

The classical computation uses $O(N^{(l)})$ bits, which are dominated by those for storing the measurement outcomes for all the $O(N^{(l)})$ qubits in the registers.

To use the measurement outcome in the other level-l gadgets, it is sufficient to guarantee that we finish calculating the measurement outcome always before starting the next level-l gadget. Indeed, our protocol performs

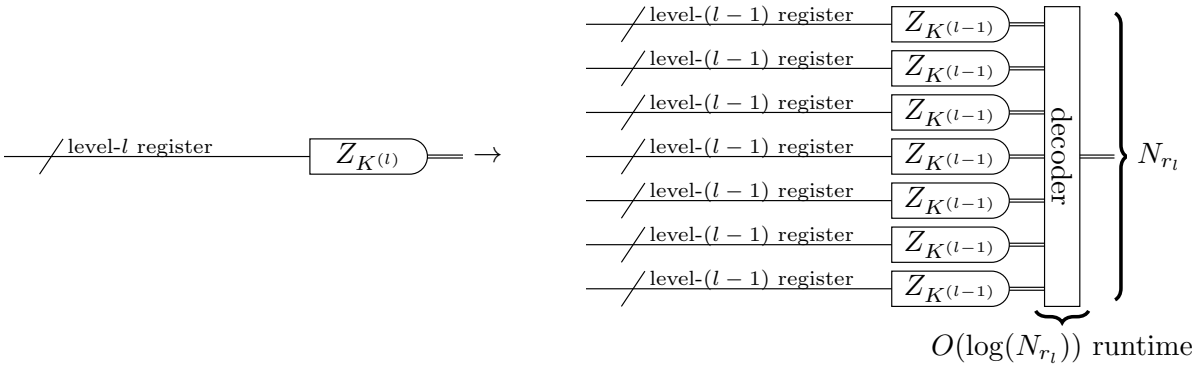


FIG. D.3. The level-l measurement gadget for performing measurements in Z basis for all the qubits in a level-l register. The gadget is implemented by transversally performing level-$(l-1)$ measurement operations, followed by the efficient decoder. Since classical computation for this decoder has $O(\log(N_{r_l}))$ runtime, we may need to perform $O(\log(N_{r_l}))$ level-$(l-1)$ wait operations before using the measurement outcome.

level-l gadgets in a synchronized way by inserting wait operations as shown in Fig. C.2. Since the runtime (D42) of the decoder is smaller than the maximum depth $G(l)$ of the level-l gadgets in (D24), the measurement outcome is always available for the next level-l gadget.

Gate gadgets: The level-l H-, CNOT-, CZ- and Pauli-gate gadgets are constructed as in Fig. D.4. The level-l H-, CNOT-, and CZ-gate gadgets are implemented by transversal level-$(l-1)$ H-, CNOT-, and CZ-gate operations, respectively, on all the level-$(l-1)$ registers. To implement the level-l Pauli-gate gadget, each level-l Pauli operator $P_{k^{(l)}}$ in (C10) is represented as a logical operator in the form of a tensor product of Pauli operators acting on the N_{r_l} level-$(l-1)$ registers, and the product of these logical operators for all $P_{k^{(l)}}$ becomes a tensor product of the Pauli operators on the N_{r_l} level-$(l-1)$ registers, i.e.,

$$\bigotimes_{n=1}^{N_{r_l}} \bigotimes_{k^{(l-1)}=1}^{K^{(l-1)}} P_{n,k^{(l-1)}}, \tag{D43}$$

where $P_{n,k^{(l-1)}} \in \{X, Z, Y, \mathbb{1}\}$ is a Pauli gate acting on the $k^{(l-1)}$th qubit in the nth level-$(l-1)$ register, and the global phase is ignored; then, the operator in (D43) is applied using the level-$(l-1)$ Pauli-gate operations transversally for all the N_{r_l} level-$(l-1)$ registers. The fault tolerance, i.e., (D17), (D18), (D19), and (D20), follows from the transversality.

The level-l H-, CNOT-, CZ-gate gadgets have a constant depth

$$O(1). \tag{D44}$$

By contrast, for the level-l Pauli-gate gadget, we need to obtain the Pauli operators (D43) from the $K^{(l)}$ Pauli operators $P_{k^{(l)}}$ by classical computation. If we can perform this classical computation during the compilation, the Pauli-gate gadget has the constant depth

$$O(1). \tag{D45}$$

On the other hand, as stated in (C19), there are cases where we need to compute (D43) on the fly while inserting wait operations in the level-l Pauli-gate gadget to wait for the runtime of this classical computation, which we bound in the following. As in the two-register Clifford-gate abbreviation in Fig. D.1, we represent the input $K^{(l)}$ Pauli operators as the $2K^{(l)}$-dimensional binary row vector. Since $\mathcal{Q}^{(l)}$ has $K^{(l-1)}$ blocks of $\mathcal{Q}_{r_l}$, we can compute the corresponding logical operator (D43) by multiplying $2K^{(l-1)}$ copies of the $K_{r_l} \times N_{r_l}$ matrices G_{r_l} in (B8) (from the right of the corresponding K_{r_l}-bit segment of this vector). Thus, using $2K^{(l-1)} \times (K_{r_l} \times N_{r_l}) = O(K^{(l)} N_{r_l}) = O(N^{(l)} N_{r_l})$ parallel processes, we can perform this matrix multiplication in time $O(\log(N_{r_l}))$. Therefore, the depth of the Pauli-gate gadget including the wait operations is

$$O(\log(N_{r_l})). \tag{D46}$$

The required number of bits is $O(N^{(l)})$, dominated by those for storing the $2K^{(l)}$-dimensional vectors.

Initial-state preparation gadget: The level-l initial-state preparation gadget is constructed as in Fig. D.5. The initial part of the gadget is aimed at implementing an encoding unitary U_{encode} to transform the initialized N_{r_l} level-$(l-1)$ registers into an encoded logical state of $|0\rangle^{\otimes K^{(l)}}$. Under the decomposition (D3) with the $K^{(l-1)}$ code blocks of $\mathcal{Q}_{r_l}$, the unitary is written in the form

$$U_{\text{encode}} = V^{\otimes K^{(l-1)}}, \tag{D47}$$

where V is an encoding unitary for the code $\mathcal{Q}_{r_l}$ satisfying

$$V(|\psi\rangle \otimes |0\rangle^{\otimes (N_{r_l} - K_{r_l})}) = |\psi_{\text{logical}}\rangle\,. \tag{D48}$$

Here, $|\psi\rangle$ on the left-hand side is any K_{r_l}-qubit state, and $|\psi_{\text{logical}}\rangle$ is the state of N_{r_l} qubits representing the logical state $|\psi\rangle$. In the circuit of Fig. D.5, N_{r_l} level-$(l-1)$ registers are initially prepared in $|0\rangle^{\otimes K^{(l-1)}}$ by the

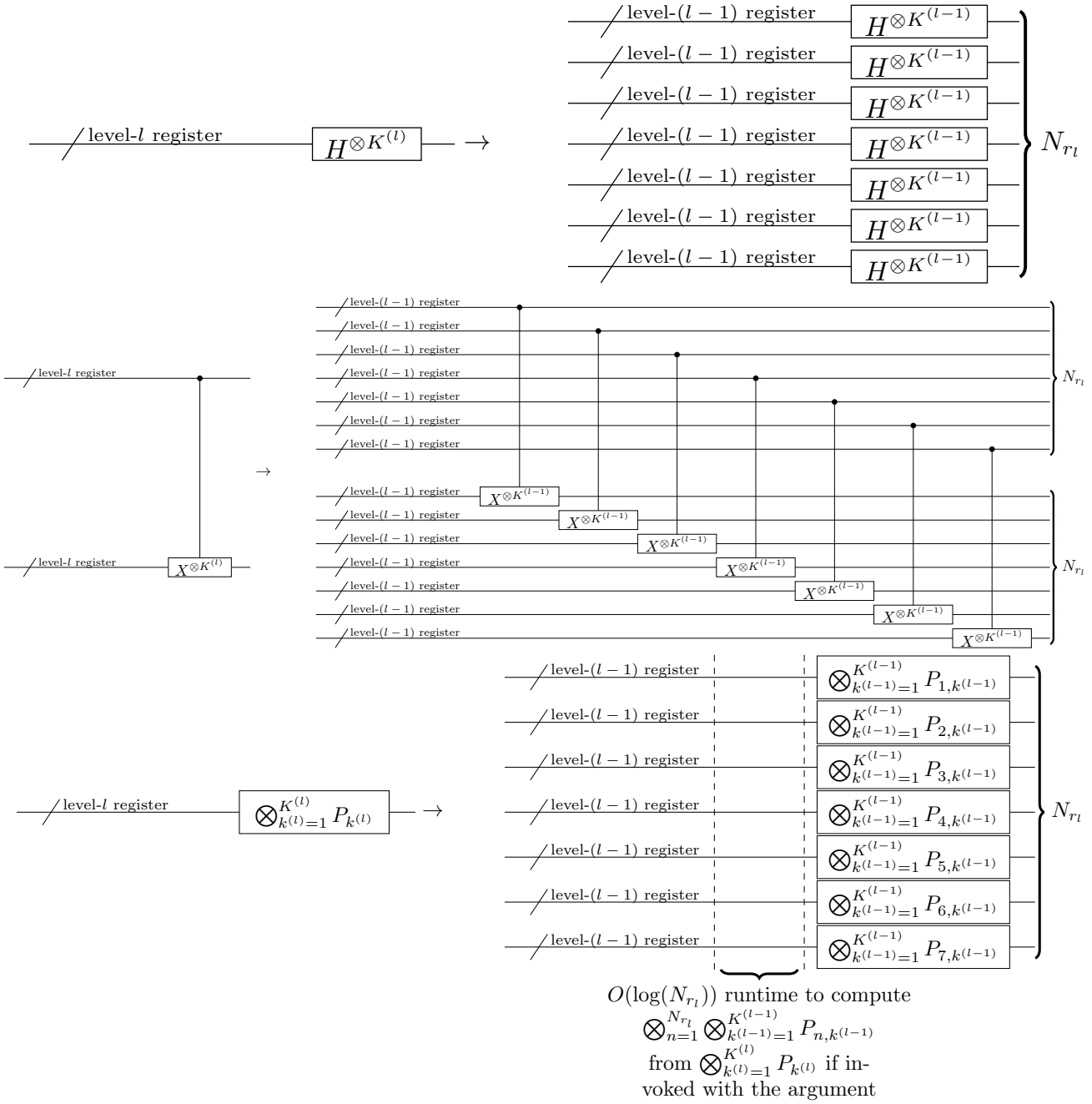


FIG. D.4. The level-l H-gate gadget for performing $H^{\otimes K^{(l)}}$ acting on all the $K^{(l)}$ qubits in a level-l register at the top, the level-l CNOT-gate gadget for performing $\mathrm{CNOT}^{\otimes K^{(l)}}$ acting on all the $K^{(l)}$ control qubits in the upper level-l register and all the $K^{(l)}$ target qubits in the lower level-l register in the middle, and the level-l Pauli-gate gadget for performing an arbitrary tensor product of Pauli gates $\bigotimes_{k^{(l)}=1}^{K^{(l)}} P_{k^{(l)}}$ for $P_{k^{(l)}} \in \{X, Z, Y, \mathbb{1}\}$ in (C10) acting on the $K^{(l)}$ qubits in a level-l register at the bottom. The level-l H- and CNOT-gate gadgets are implemented by level-$(l-1)$ H- and CNOT-gate operations, respectively, acting transversally on all the level-$(l-1)$ registers. In the same way as the level-l CNOT-gate gadget but using CZ in place of CNOT, the level-l CZ-gate gadget for performing $CZ^{\otimes K^{(l)}}$ acting on two level-l registers is defined. The level-l Pauli-gate gadget can be implemented by performing Pauli gates $\bigotimes_{n=1}^{N_{r_l}} \bigotimes_{k^{(l-1)}=1}^{K^{(l-1)}} P_{n,k^{(l-1)}}$ in (D43), i.e., by applying level-$(l-1)$ Pauli-gate operations transversally to the N_{r_l} level-$(l-1)$ registers. Note that the $O(\log(N_{r_l}))$-time classical computation shown in the Pauli-gate gadget can be performed during the compilation prior to starting execution of the quantum computation, except for the cases where the gadget is invoked with the argument to specify the gate.

level-$(l-1)$ initial-state preparation operations in (C6), which $U_{\rm encode}$ transforms into logical $|0\rangle^{\otimes K^{(l)}}$.

For any stabilizer code, e.g., the code used here, the unitary $U_{\rm encode}$ for encoding is Clifford; in particular, to implement $U_{\rm encode}$, the analysis here uses the fact shown in Ref. [37] that an encoding unitary V for the stabilizer code with N_{r_l} physical qubits and K_{r_l} logical qubits can be implemented by an N_{r_l}-qubit stabilizer circuit using $O(N_{r_l}(N_{r_l}-K_{r_l}))=O(N_{r_l}\log(N_{r_l}))$ one- and two-qubit Clifford gates, which lead to the depth at most $O(N_{r_l}\log(N_{r_l}))$. Then a circuit for $U_{\rm encode}$ is constructed by replacing each gate G in the stabilizer circuit for V with $G^{\otimes K^{(l-1)}}$ according to (D47). The stabilizer circuit for V constructed by the technique in Ref. [37] is composed only of H, CNOT, CZ, and Z (i.e., Pauli) gates. Thus, in the stabilizer circuit, we use the level-$(l-1)$ gate operations corresponding to these gates; that is, we do not need to use the two-register Clifford-gate abbreviation in Fig. D.1. As a result, $U_{\rm encode}$ is implemented by a level-$(l-1)$ circuit with depth $O(N_{r_l}\log(N_{r_l}))$.

The level-$(l-1)$ stabilizer circuit for the encoding by itself is non-fault-tolerant in the sense that one fault among the level-$(l-1)$ locations of the circuit may lead to an uncorrectable error at the end of the circuit; to remedy this, verification is performed by measuring the logical Z operators of all the logical qubits, and all the Z generators of the stabilizer, for each of the $K^{(l-1)}$ code blocks of $\mathcal{Q}_{r_l}$. To measure these operators without destroying the state itself, another set of N_{r_l} level-$(l-1)$ registers are prepared in logical $|0\rangle^{\otimes K^{(l)}}$ by the same non-fault-tolerant level-$(l-1)$ stabilizer circuit, and all the syndromes are extracted at once by the constant-depth CNOT-gate operations in the level-$(l-1)$ circuit of Fig. D.5. We obtain each syndrome from the measurement outcomes by classical computation using the description of the logical Z operators and the Z stabilizer generators.

If no error is detected from these syndromes, i.e., in the case of success in the verification, then the gadget outputs the logical $|0\rangle^{\otimes K^{(l)}}$ prepared in this first run. Otherwise, i.e., if the verification in this first run detects an error, then the logical $|0\rangle^{\otimes K^{(l)}}$ prepared in this first run is discarded. This ensures that a fault in a single location leading to incorrigible multiple X errors is always detected and discarded. Note that since $|0\rangle$ is stabilized by Z, i.e., $Z|0\rangle=|0\rangle$, detection of Z errors is unnecessary for verifying $|0\rangle$; that is, multiple Z errors before the ideal decoder in (D12) may lead to a logical Z error, but do not change the decoded state. If the first run has discarded the state, the gadget performs the same non-fault-tolerant level-$(l-1)$ stabilizer circuit for the preparation and outputs the logical $|0\rangle^{\otimes K^{(l)}}$ prepared in the second run without verification. In this case, a single fault should be present in the part of the first run, and thus the part of the preparation in the second run has no fault as long as the gadget has at most one faulty location. Therefore, in any of these two cases, the gadget satisfies the conditions (D11) and (D12) of fault tolerance.

The depth of the level-$(l-1)$ circuit for implementing the initial-state preparation gadget is dominated by the encoding part for implementing $U_{\rm encode}$. The depth of the gadget including both the first and second runs is at most twice as long as the first run, and thus in the following, we analyze the depth of the first run. The required number of H-, CNOT-, CZ-, and Pauli-gate operations in the encoding part is $O\left(N_{r_l}\log(N_{r_l})\right)$, which is an upper bound of the depth of this part. As for the other parts, the $O(N_{r_l})$ preparations of $|0\rangle^{\otimes K^{(l-1)}}$ in the gadget have the depth $O(N_{r_l})$. Since $\mathcal{Q}^{(l)}$ has $K^{(l-1)}$ blocks of $\mathcal{Q}_{r_l}$, the classical computation for the post-selection computes the inner product between the $(N_{r_l}\times K^{(l-1)})$-bit measurement outcome and each row of $K^{(l-1)}$ copies of H_{r_l} and G_{r_l} in (B1) and (B8), respectively. Here, H_{r_l} has r_l rows, and G_{r_l} has K_{r_l} rows. Thus, using $(r_l+K_{r_l})\times(N_{r_l}\times K^{(l-1)})=O(N_{r_l}N^{(l)})$ parallel processes (where we use $K_{r_l}K^{(l-1)}=K^{(l)}=O(N^{(l)})$), we can perform all the inner products in runtime $O(\log(N_{r_l}))$. At this point, we have the $(r_l+K_{r_l})\times K^{(l-1)}$-bit string representing these inner products. By taking bitwise OR over this bit string by parallel processes, we check whether the bit string includes 1 within the runtime of $O(\log((r_l+K_{r_l})\times K^{(l-1)}))=O(\log(N^{(l)}))$. Consequently, the depth of the initial-state preparation gadget is

$$O\left(N_{r_l}\log(N_{r_l})\right). \tag{D49}$$

The required number of bits is $O(N_{r_l}\times K^{(l-1)})=O(N^{(l)})$, dominated by those for storing and processing the $(N_{r_l}\times K^{(l-1)})$-bit measurement outcome.

Note that in place of the stabilizer circuit for the encoding unitary V in Ref. [37], Refs. [38, 39] provide more optimized stabilizer circuits for preparing logical $|0\rangle$, which could also be used here. Moreover, the depth of a stabilizer circuit could be shortened by parallelizing the circuit using auxiliary qubits [40, 41], but such further optimization is not considered here. Our analysis of the existence of a threshold and the space and time overheads holds without such an optimization.

Error-correction gadget: The level-l error-correction gadget is constructed as in Fig. D.6 based on Knill's error correction [34, 35]. The gadget is assisted by $2(N_{r_l}+1)$ level-$(l-1)$ registers. Using the same circuit as the initial-state preparation gadget in Fig. D.5 on these $2(N_{r_l}+1)$ level-$(l-1)$ registers, we prepare N_{r_l} out of these $2(N_{r_l}+1)$ in logical $|0\rangle^{\otimes K^{(l)}}$. Using this logical $|0\rangle^{\otimes K^{(l)}}$, the circuit in Fig. D.6 conducts quantum teleportation of the state of the encoded logical level-l register. The $2K^{(l)}$-bit outcome from the measurement gadgets is fed to an efficient decoder to calculate a logical operator used as correction in quantum teleportation [36]

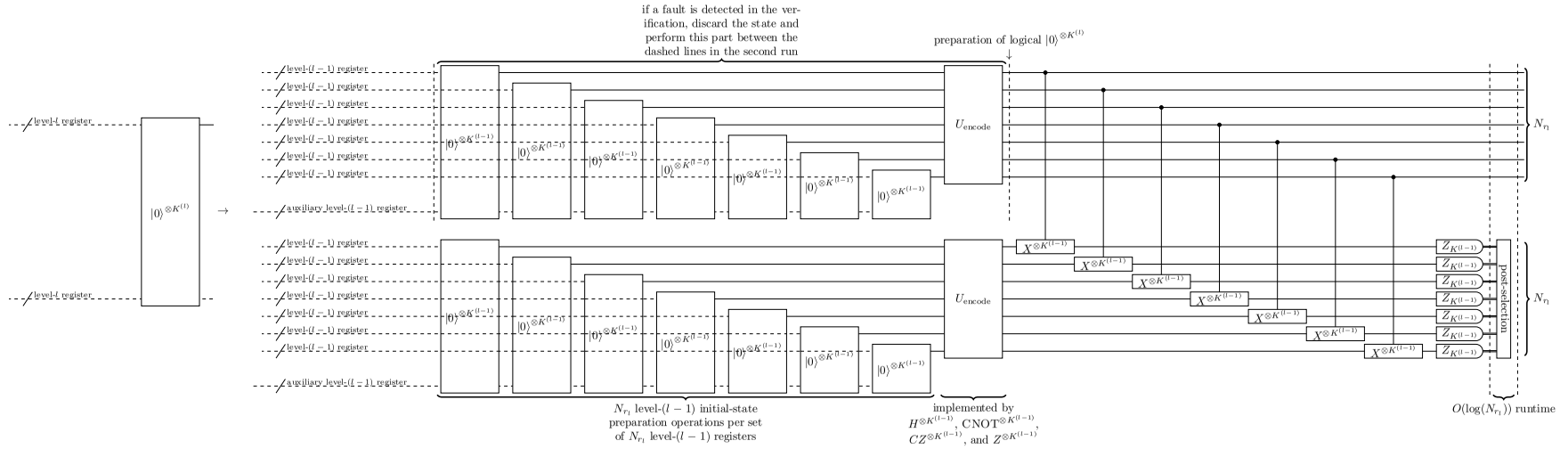


FIG. D.5. The level-l initial-state preparation gadget for preparing a logical state $|0\rangle^{\otimes K^{(l)}}$ in an encoded level-l register. The gadget starts with a level-$(l-1)$ stabilizer circuit for preparing logical $|0\rangle^{\otimes K^{(l)}}$ by implementing a Clifford unitary U_{encode} in (D47) for encoding. In this level-$(l-1)$ stabilizer circuit, preparation operations are performed sequentially using the auxiliary level-$(l-1)$ registers added to each set of N_{r_l} level-$(l-1)$ registers on which the operations themselves act. Note that most of the auxiliary level-$(l-1)$ registers are omitted in the figure except for those used for these preparations. Then, verification is performed by measuring the logical Z operator for each of the logical qubits of $\mathcal{Q}_{r_l}$, and all the Z generators of the stabilizer of $\mathcal{Q}_{r_l}$, for each of the $K^{(l-1)}$ code blocks. If no error is detected in the verification, the gadget outputs the logical $|0\rangle^{\otimes K^{(l)}}$ prepared in this first run; otherwise, this prepared state is discarded, and the gadget performs the part between the dashed lines as shown in the figure and outputs the logical $|0\rangle^{\otimes K^{(l)}}$ prepared in the second run without verification.

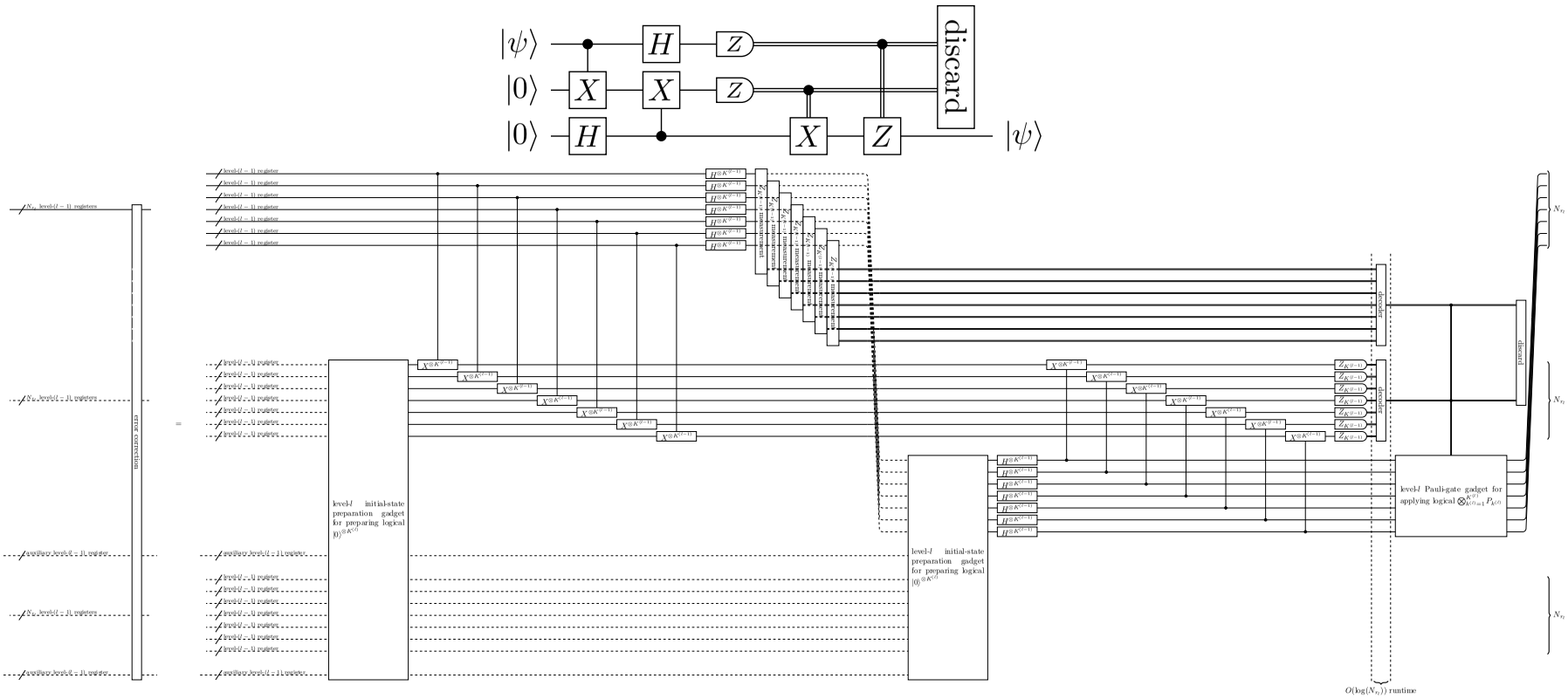


FIG. D.6. A circuit for Knill's error correction at the top and the level-l error-correction gadget based on Knill's error correction at the bottom. The gadget is assisted by $2(N_{r_l}+1)$ level-$(l-1)$ registers for preparing logical $|0\rangle^{\otimes K^{(l)}}$. The white boxes in the figure for the level-l initial-state preparation gadgets and the level-l Pauli-gate gadgets are to be replaced with the corresponding level-$(l-1)$ circuits shown on the right-hand sides of Figs. D.5 and D.4, respectively. We calculate the measurement outcome in the same way as the measurement gadget in Fig. D.3. Given the measurement outcome, we calculate a logical operator $\bigotimes_{k^{(l)}=1}^{K^{(l)}} P_{k^{(l)}}$ in (D50) in the form of a tensor product of Pauli gates $P_{k^{(l)}}$ according to correction in quantum teleportation. In the level-$(l-1)$ circuit, logical $\bigotimes_{k^{(l)}=1}^{K^{(l)}} P_{k^{(l)}}$ for the correction in the quantum teleportation is implemented by invoking the level-l Pauli-gate gadgets in Fig. D.4 with an argument to specify the gate.

$$\bigotimes_{k^{(l)}=1}^{K^{(l)}} P_{k^{(l)}}, \tag{D50}$$

where $P_{k^{(l)}} \in \{X, Z, Y, \mathbb{1}\}$ is a Pauli gate acting on the $k^{(l)}$th qubit in the level-l register with labeling (B23). As in the level-l measurement gadget, the measurement in this quantum teleportation needs wait operations to wait for the $O(\log(N_{r_l}))$ runtime of classical computation. This logical operator is implemented by invoking the level-l Pauli-gate gadget in Fig. D.4 with the argument. With no fault, this gadget carries out error correction whereas the fault tolerance follows from the transversality, satisfying (D21) and (D22).

The depth of the level-$(l-1)$ circuit is

$$O(N_{r_l} \log(N_{r_l})), \tag{D51}$$

dominated by the depth (D49) of the level-l initial-state preparation gadgets. As for the other parts, the runtime of classical computation for decoders from the measurement outcomes is $O(\log(N_{r_l}))$ as discussed above, and that for the level-l Pauli-gate gadget is also $O(\log(N_{r_l}))$. All the other parts have a constant depth. The required number of bits is dominated by $O(N^{(l)})$ to store and process the measurement outcome.

Clifford-state preparation gadget: The level-l Clifford-state preparation gadget is constructed as in Figs. D.7 and D.8. The gadget starts with a level-$(l-1)$ stabilizer circuit for preparing logical $(\mathbb{1}^{B_1B_2} \otimes U_{\rm C}^{B_3B_4}) |\Phi^{(l)}\rangle^{B_1B_2B_3B_4}$, where $|\Phi^{(l)}\rangle^{B_1B_2B_3B_4}$ is the maximally entangled state between B_1B_2 and B_3B_4 given by (C11). Each of the level-l registers B_1, B_2, B_3, B_4 is encoded in N_{r_l} level-$(l-1)$ registers in the level-$(l-1)$ circuit, and these N_{r_l} level-$(l-1)$ registers will also be collectively called B_1, B_2, B_3, B_4, respectively. After preparing logical $|\Phi^{(l)}\rangle^{B_1B_2B_3B_4}$, application of logical $U_{\rm C}^{B_3B_4}$ is implemented by using the level-$(l-1)$ elementary operations in the following way. Using the Clifford unitary $U_{\rm encode}$ in (D47), logical $U_{\rm C}$ can be implemented by a Clifford unitary

$$\begin{aligned}&(U_{\rm encode}^{B_3} \otimes U_{\rm encode}^{B_4})\times\\ &(U_{\rm C} \otimes \mathbb{1}^{\otimes 2(N_{r_l}K^{(l-1)}-K^{(l)})})\times\\ &(U_{\rm encode}^{B_3} \otimes U_{\rm encode}^{B_4})^\dagger,\end{aligned} \tag{D52}$$

where $U_{\rm C}$ acts on the $2K^{(l)}$ qubits that hold the decoded state of B_3 and B_4, and $\mathbb{1}^{\otimes 2(N_{r_l}K^{(l-1)}-K^{(l)})}$ is the identity operator acting on the remaining $2(N_{r_l}-K_{r_l})K^{(l-1)} = 2(N_{r_l}K^{(l-1)}-K^{(l)})$ qubits in the $2N_{r_l}$ level-$(l-1)$ registers for B_3 and B_4. We decompose $U_{\rm encode}$ into elementary operations as discussed for the initial-state preparation gadget in Fig. D.5 and also decompose $U_{\rm C}$ into elementary operations, which we will discuss later when we evaluate the depth of the gadget.

Similar to the encoding in the initial-state preparation gadget of Fig. D.5, the level-$(l-1)$ stabilizer circuit for preparing logical $(\mathbb{1}^{B_1B_2} \otimes U_{\rm C}^{B_3B_4}) |\Phi^{(l)}\rangle^{B_1B_2B_3B_4}$ by itself is non-fault-tolerant in the sense that one fault among level-$(l-1)$ locations in the circuit may lead to an uncorrectable error at the end of the circuit, which necessitates a part for performing verification. The verification part starts with applying the level-l error-correction gadgets in Fig. D.6 to B_1, B_2, B_3, and B_4, so as to ensure that the state is in the code space of $\mathcal{Q}_{r_l}$ regardless of faults in the preparation part. The circuit is followed by measurements of the logical stabilizer operators of logical $(\mathbb{1}\otimes U_{\rm C}) |\Phi^{(l)}\rangle$ to detect logical errors with post-selection. There are $4K^{(l)}$ stabilizer operators of $(\mathbb{1}\otimes U_{\rm C}) |\Phi^{(l)}\rangle$, i.e.,

$$\begin{aligned}&(X_{j,k^{(l)}} \otimes \mathbb{1}^{\otimes(2K^{(l)}-1)})^{B_1B_2} \otimes\\ &\quad (U_{\rm C}(X_{j+2,k^{(l)}} \otimes \mathbb{1}^{\otimes(2K^{(l)}-1)})U_{\rm C}^\dagger)^{B_3B_4},\end{aligned} \tag{D53}$$

$$\begin{aligned}&(Z_{j,k^{(l)}} \otimes \mathbb{1}^{\otimes(2K^{(l)}-1)})^{B_1B_2} \otimes\\ &\quad (U_{\rm C}(Z_{j+2,k^{(l)}} \otimes \mathbb{1}^{\otimes(2K^{(l)}-1)})U_{\rm C}^\dagger)^{B_3B_4},\end{aligned} \tag{D54}$$

where $j \in \{1,2\}$, and $k^{(l)} \in \{1, \ldots, K^{(l)}\}$. In (D53) and (D54), $X_{j,k^{(l)}}$ and $Z_{j,k^{(l)}}$ are the logical X and Z operators, respectively, of the $k^{(l)}$th logical qubits in B_j for $j \in \{1,2,3,4\}$, and $\mathbb{1}^{\otimes(2K^{(l)}-1)}$ is the identity operator acting on the rest of logical qubits.

To measure the $2K^{(l)}$ logical operators in (D53) without destroying the state itself, we extract the syndromes with the circuit shown in Fig. D.8, using $2K_{r_l}$ auxiliary level-$(l-1)$ registers twice. Note that the measurement of the other $2K^{(l)}$ logical operators (D54) is performed in the same way as (D53) using Z in place of X, and hence the following explanation will focus on (D53). The unitary gates $U_X^{A_1B_1}$, $U_X^{A_1B_2}$, $U_{X,C,3}^{A_1B_3B_4}$, and $U_{X,C,4}^{A_1B_3B_4}$ used in the circuit of Fig. D.8 will be defined later in (D63) and (D64). Among the $2K_{r_l}$ auxiliary level-$(l-1)$ registers, the first K_{r_l} level-$(l-1)$ registers are collectively denoted by A_1, and the second by A_2. Each of these $2K_{r_l}$ auxiliary level-$(l-1)$ registers is prepared in the state $|0\rangle^{\otimes K^{(l-1)}}$ by the level-$(l-1)$ initial-state preparation operation in (C6). Each of A_1 and A_2 has $K_{r_l} \times K^{(l-1)} = K^{(l)}$ qubits in its K_{r_l} level-$(l-1)$ registers. In each of the two uses of the K_{r_l} auxiliary level-$(l-1)$ registers of A_1, we measure $K^{(l)}$ out of the $2K^{(l)}$ logical operators in (D53), where a logical error can be detected as a bit flip of the measurement outcomes in A_1. The other K_{r_l} auxiliary level-$(l-1)$ registers of A_2 are used as flag qubits to make these syndrome measurements fault-tolerant [42], where an error on A_1 that may propagate and cause a logical error is detected as a bit flip of the measurement outcomes in A_2. The flip of each of the $2K^{(l)}$-bit measurement outcomes can be checked using $O(N^{(l)})$ parallel processes immediately, i.e., in runtime $O(1)$.

If no error is detected either from the $2K^{(l)}$ logical stabilizer operators in (D53) and the flag qubits in this verifi-

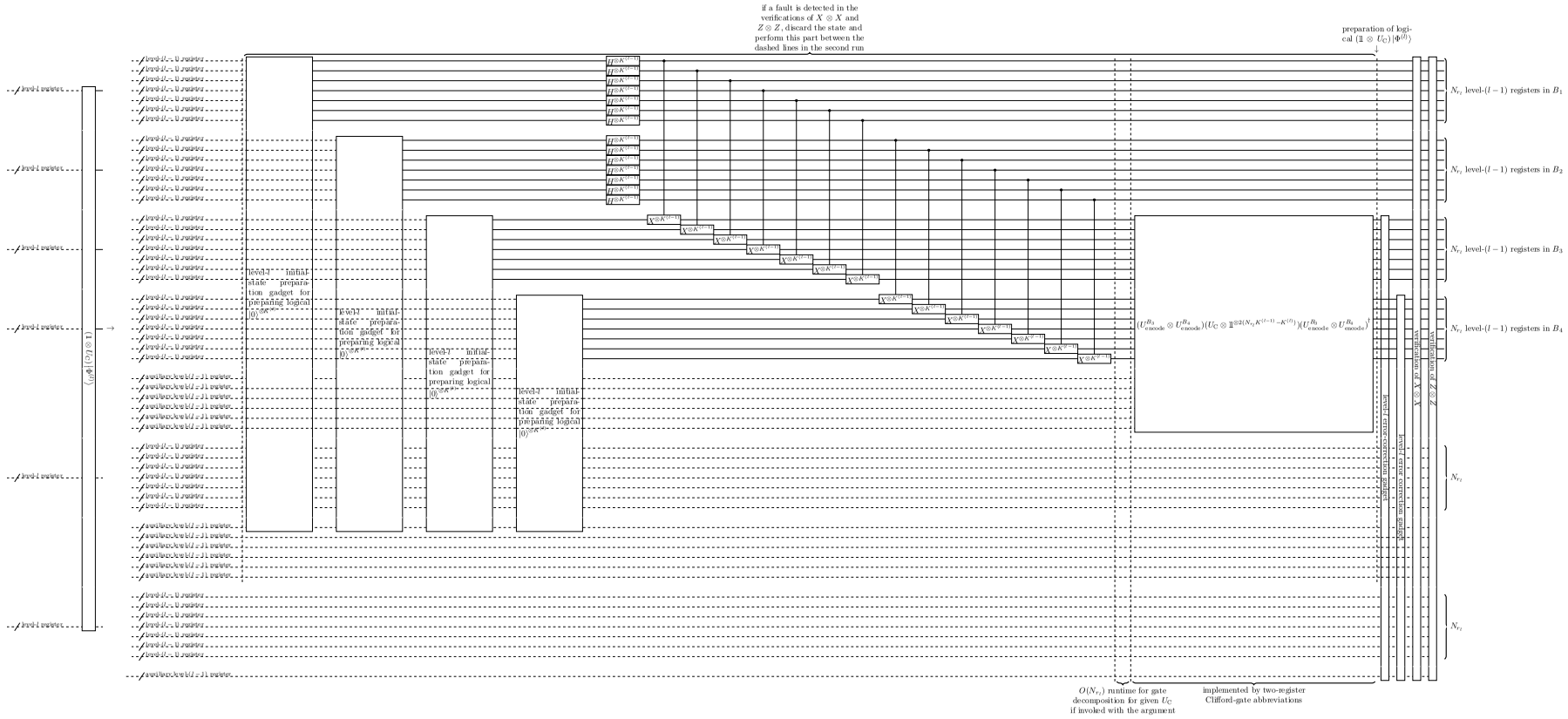


FIG. D.7. The level-l Clifford-state preparation gadget for preparing a logical state $(\mathbb{1}^{B_1B_2} \otimes U_C^{B_3B_4})|\Phi^{(l)}\rangle^{B_1B_2B_3B_4}$, where each of B_1, B_2, B_3, B_4 is N_{r_l} level-$(l-1)$ registers for an encoded level-l register aligned from top to bottom in the level-$(l-1)$ circuit. The white boxes of the level-l initial-state preparation gadgets and the level-l error-correction gadgets are replaced with the level-$(l-1)$ circuits on the right-hand side of Figs. D.5 and D.6, respectively. The white boxes of verification of $X \otimes X$ and $Z \otimes Z$ are replaced with the level-$(l-1)$ circuit on the right-hand side of Fig. D.8 and the same level-$(l-1)$ circuit with substituting X with Z, respectively. The gadget starts with a level-$(l-1)$ stabilizer circuit for preparing logical $(\mathbb{1} \otimes U_C)|\Phi^{(l)}\rangle$ by implementing (D52). Then, verification is performed by using the level-l error-correction gadgets in Fig. D.6 so as to ensure that the state is in the code space of $\mathcal{Q}_{r_l}$, followed by measuring the logical stabilizer operators of logical $(\mathbb{1} \otimes U_C)|\Phi^{(l)}\rangle$, i.e., all the logical operators in (D53) and (D54), to detect logical errors. A post-selection is made according to the verifications of $X \otimes X$ and $Z \otimes Z$. If no error is detected in these verifications, the gadget outputs the logical $(\mathbb{1}^{B_1B_2} \otimes U_C^{B_3B_4})|\Phi^{(l)}\rangle^{B_1B_2B_3B_4}$ prepared in this first run; otherwise, this prepared state is discarded, and the gadget again performs the part between the dashed lines as shown in the figure and outputs the logical $(\mathbb{1}^{B_1B_2} \otimes U_C^{B_3B_4})|\Phi^{(l)}\rangle^{B_1B_2B_3B_4}$ prepared in this second run without verification. Note that classical computation for the gate decomposition of the gadget can be performed during the compilation prior to starting execution of the quantum computation, except for the cases where the gadget is invoked with the argument to specify the gate as discussed with (D88).

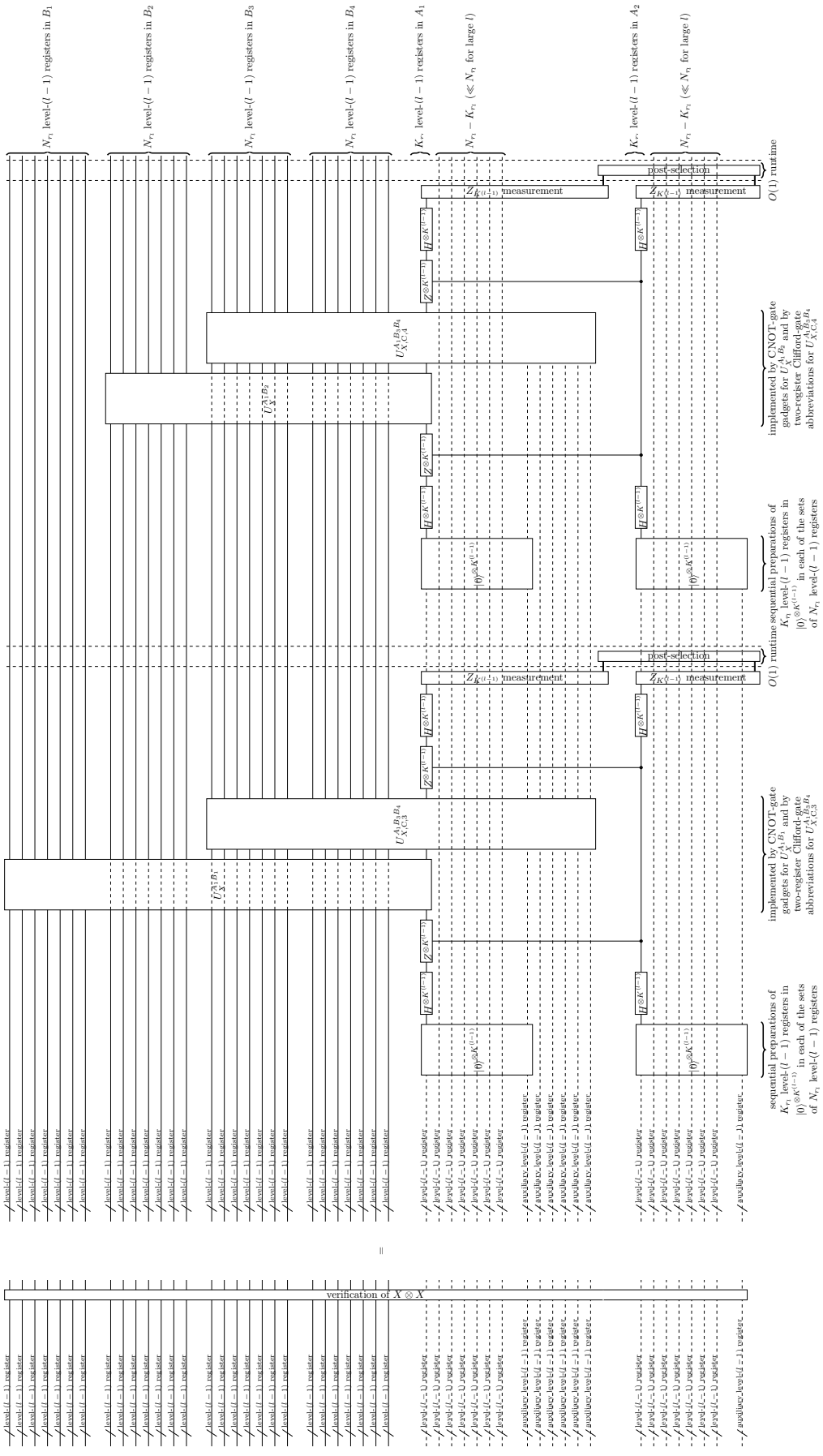


FIG. D.8. A circuit for measuring the logical stabilizer operators of logical $(\mathbb{1}^{B_1B_2} \otimes U_C^{B_3B_4}) \left|\Phi^{(l)}\right\rangle^{B_1B_2B_3B_4}$. The circuit for extracting all the logical operators in (D53) to a set of K_{r_l} level-$(l-1)$ registers is performed by controlled Pauli gates $U_X^{A_1B_1}$, $U_X^{A_2B_2}$, $U_{X,C}^{A_3B_3}$, and $U_{X,C}^{A_4B_4}$ defined as (D63) and (D64), while that for measuring all the logical operators in (D54) is the same as this circuit with substituting X with Z. In extracting these logical operators, another set of K_{r_l} level-$(l-1)$ registers are used as flag qubits to make the verification fault-tolerant. The state is discarded unless no error is detected either from these logical stabilizer operators in (D53) and the flag qubits.

cation of $X\otimes X$, and also either from the $2K^{(l)}$ logical stabilizer operators in (D54) and the flag qubits in the subsequent verification of $Z\otimes Z$, then the gadget outputs the logical $(\mathbb{1}^{B_1B_2}\otimes U_{\rm C}^{B_3B_4})\left|\Phi^{(l)}\right\rangle^{B_1B_2B_3B_4}$ prepared in this first run. Otherwise, the gadget discards the state prepared in the first run and performs the same non-fault-tolerant level-$(l-1)$ stabilizer circuit for the preparation and outputs the logical $(\mathbb{1}^{B_1B_2}\otimes U_{\rm C}^{B_3B_4})\left|\Phi^{(l)}\right\rangle^{B_1B_2B_3B_4}$ prepared in the second run without verification.

The Clifford-state preparation gadget becomes fault-tolerant, that is, satisfies the conditions (D13) and (D14). Since the case with no fault ($s=0$) is obvious, here we explain the reason why (D14) with $s=1$ holds true. The gadget can have only the first run (with the success in the verification) or both the first and second runs. Whenever the gadget needs to perform the second run, the single fault should be present in the part of the first run; then, the output state in the second run has no error. Thus, in the following, we consider the cases where the gadget has only the first run without detecting any error in the verification. The cases with $s=1$ belong to either of the following two cases: (i) a case where the single fault occurs on one of the level-$(l-1)$ locations in the part of the level-$(l-1)$ circuit in Fig. D.7 for preparing logical $(\mathbb{1}^{B_1B_2}\otimes U_{\rm C}^{B_3B_4})\left|\Phi^{(l)}\right\rangle^{B_1B_2B_3B_4}$, and (ii) a case where the single fault occurs on one of the level-$(l-1)$ locations in the part of the level-$(l-1)$ circuit in Fig. D.7 for the verification. In case (i), since the error-correction gadgets have no fault, these gadgets leave B_1 through B_4 in a state in the code space of $4K^{(l)}$ logical qubits; in addition, the measurements of the $4K^{(l)}$ logical stabilizer operators in (D53) and (D54) are also faultless, and thus finding no logical error specifies the logical state of all the $4K^{(l)}$ logical qubits as $(\mathbb{1}\otimes U_{\rm C})\left|\Phi^{(l)}\right\rangle$ in the code space. As for case (ii), the prepared state before the verification is the logical state $(\mathbb{1}^{B_1B_2}\otimes U_{\rm C}^{B_3B_4})\left|\Phi^{(l)}\right\rangle^{B_1B_2B_3B_4}$ without any error, and the fault in the verification does not cause any uncorrectable error at the end of the level-$(l-1)$ circuit. In particular, a fault in the error-correction gadgets does not cause any uncorrectable error owing to transversality. On the other hand, in the measurement of the logical operators, an error on the registers in A_1 may propagate and spread, leading to uncorrectable errors. Nonetheless, this fault causes no problem because such an error is always detected by the measurements on the registers A_2, thanks to the technique of flag qubits [42]. Finally, there may be a question of whether a correctable error on B_1,B_2,B_3,B_4 would be aggravated by the subsequent faultless measurement of the logical operators, but since the bit-flip or phase-flip error propagating from B_1,B_2,B_3,B_4 to A_1,A_2 never propagates back from A_1,A_2 to B_1,B_2,B_3,B_4 by construction, the error remains correctable. These two cases show fault tolerance.

The unitary gates $U_X^{A_1B_1}$, $U_X^{A_1B_2}$, $U_{X,\mathrm{C},3}^{A_1B_3B_4}$, and $U_{X,\mathrm{C},4}^{A_1B_3B_4}$ used for the measurements of the logical operators in (D53) are defined as follows, where we may omit the identity operator $\mathbb{1}$ for simplicity of presentation. In the following, we may use the mapping

$$k^{(l)}\mapsto(k,k^{(l-1)}) \tag{D55}$$

defined as (B23) for simplicity of notation. For each $k^{(l)}\in\{1,\dots,K^{(l)}\}$, let $U_{X,k^{(l)}}^{B_1}$ denote an operator representing the logical operator $X_{1,k^{(l)}}$ appearing in (D53) for the $k^{(l)}$th logical qubit, in the form of a tensor product of X and $\mathbb{1}$ acting on the $N_{r_l}K^{(l-1)}$ qubits of the N_{r_l} level-$(l-1)$ registers that form B_1; under the mapping (D55), we can write $U_{X,k^{(l)}}^{B_1}$ as

$$U_{X,k^{(l)}}^{B_1}=\prod_{n=1}^{N_{r_l}}P_{X,k,n}^{B_1,n,k^{(l-1)}} \tag{D56}$$

where

$$P_{X,k,n}\in\{X,\mathbb{1}\} \tag{D57}$$

is determined in such a way that $\bigotimes_{n=1}^{N_{r_l}}P_{X,k,n}$ should be the logical X operator of the kth logical qubit of $\mathcal{Q}_{r_l}$, and the superscript of $P_{X,k,n}^{B_1,n,k^{(l-1)}}$ shows that $P_{X,k,n}^{B_1,n,k^{(l-1)}}$ acts on the $k^{(l-1)}$th qubit in the nth level-$(l-1)$ register of B_1. The unitaries $U_{X,k^{(l)}}$ for all $k^{(l)}\in\{1,\dots,K^{(l)}\}$ commute with each other since each $U_{X,k^{(l)}}$ is a tensor product only of X and $\mathbb{1}$. Similarly, let $U_{X,\mathrm{C},3,k^{(l)}}^{B_3B_4}$ denote an operator representing the logical operator $(U_{\rm C}(X_{3,k^{(l)}}\otimes\mathbb{1}^{\otimes(2K^{(l)}-1)})U_{\rm C}^\dagger)^{B_3B_4}$ in (D53). Since $U_{\rm C}$ is Clifford, $U_{X,\mathrm{C},3,k^{(l)}}^{B_3B_4}$ can be written under the mapping (D55) as a tensor product of Pauli operators

$$\begin{aligned}U_{X,\mathrm{C},3,k^{(l)}}^{B_3B_4}=&\\ &\prod_{n=1}^{N_{r_l}}\prod_{k^{(l-1)\prime}=1}^{K^{(l-1)}}\Big(P_{X,3,3,k^{(l-1)},k,n,k^{(l-1)\prime}}^{B_3,n,k^{(l-1)\prime}}\otimes\\ &\quad P_{X,3,4,k^{(l-1)},k,n,k^{(l-1)\prime}}^{B_4,n,k^{(l-1)\prime}}\Big),\end{aligned} \tag{D58}$$

where

$$P_{X,3,j,k^{(l-1)},k,n,k^{(l-1)\prime}}\in\{\pm X,\pm Z,\pm Y,\pm\mathbb{1}\} \tag{D59}$$

for $j\in\{3,4\}$, and $P_{X,3,j,k^{(l-1)},k,n,k^{(l-1)\prime}}^{B_j,n,k^{(l-1)\prime}}$ acts on the $k^{(l-1)\prime}$th qubit in the nth level-$(l-1)$ register of B_j. The unitaries $U_{X,\mathrm{C},3,k^{(l)}}$ for all $k^{(l)}\in\{1,\dots,K^{(l)}\}$ commute with each other due to the commutativity of the logical operators $U_{\rm C}(X_{3,k^{(l)}}\otimes\mathbb{1}^{\otimes(2K^{(l)}-1)})U_{\rm C}^\dagger$.

Using the operators (D56) and (D58) given by the tensor product of Pauli operators, controlled Pauli gates used for measuring (D53) are defined as follows. For a Pauli operator $P^{B_j,n,k^{(l-1)\prime}}$ used in (D56) and (D58) (where $k^{(l-1)\prime}=k^{(l-1)}$ for (D56)), let

$$C^{A_1,k,k^{(l-1)}}P^{B_j,n,k^{(l-1)\prime}} \tag{D60}$$

denote a controlled Pauli operator that applies $P^{B_j,n,k^{(l-1)\prime}}$ to the target qubit, i.e., the $k^{(l-1)\prime}$th qubit in the nth level-$(l-1)$ register of B_j, controlled by the $k^{(l-1)}$th qubit in the kth level-$(l-1)$ register of A_1. Define $CU^{A_1B_1}_{X,k^{(l)}}$ as a controlled unitary operator given under the mapping (D55) by

$$CU^{A_1B_1}_{X,k^{(l)}} = \prod_{n=1}^{N_{r_l}} C^{A_1,k,k^{(l-1)}} P^{B_1,n,k^{(l-1)}}_{X,k,n}. \tag{D61}$$

The controlled unitaries $CU_{X,k^{(l)}}$ for all $k^{(l)} \in \{1,\dots,K^{(l)}\}$ commute with each other due to the commutativity of $U_{X,k^{(l)}}$. Define $CU^{A_1B_3B_4}_{X,\mathrm{C},3,k^{(l)}}$ in the same way as $CU^{A_1B_1}_{X,k^{(l)}}$ using $U^{B_3B_4}_{X,\mathrm{C},3,k^{(l)}}$ in place of $U^{B_1}_{X,k^{(l)}}$; that is, $CU^{A_1B_3B_4}_{X,\mathrm{C},3,k^{(l)}}$ is given under the mapping (D55) by

$$\begin{aligned} &CU^{A_1B_3B_4}_{X,\mathrm{C},3,k^{(l)}} = \\ &\prod_{n=1}^{N_{r_l}} \prod_{k^{(l-1)\prime}=1}^{K^{(l-1)}} \Big(C^{A_1,k,k^{(l-1)}} P^{B_3,n,k^{(l-1)\prime}}_{X,3,3,k^{(l-1)},k,n,k^{(l-1)\prime}} \otimes \\ &\quad C^{A_1,k,k^{(l-1)}} P^{B_4,n,k^{(l-1)\prime}}_{X,3,4,k^{(l-1)},k,n,k^{(l-1)\prime}} \Big). \end{aligned} \tag{D62}$$

The controlled unitaries $CU_{X,\mathrm{C},3,k^{(l)}}$ for all $k^{(l)} \in \{1,\dots,K^{(l)}\}$ commute with each other due to the commutativity of $U_{X,\mathrm{C},3,k^{(l)}}$. Define $CU^{A_1B_3B_4}_{X,\mathrm{C},4,k^{(l)}}$ in the same way as $CU^{A_1B_3B_4}_{X,\mathrm{C},3,k^{(l)}}$ using $X_{4,k^{(l)}}$ in place of $X_{3,k^{(l)}}$. Then, define $U^{A_1B_1}_X$ in Fig. D.8 as the product of commuting operators $CU^{A_1B_1}_{X,k^{(l)}}$ for all $k^{(l)} \in \{1,\dots,K^{(l)}\}$, i.e., under the mapping (D55),

$$\begin{aligned} U^{A_1B_1}_X &\coloneqq \prod_{k^{(l)}=1}^{K^{(l)}} CU^{A_1B_1}_{X,k^{(l)}} \\ &= \prod_{k=1}^{K_{r_l}} \prod_{n=1}^{N_{r_l}} \left(\prod_{k^{(l-1)}=1}^{K^{(l-1)}} C^{A_1,k,k^{(l-1)}} P^{B_1,n,k^{(l-1)}}_{X,k,n} \right), \end{aligned} \tag{D63}$$

and $U^{A_1B_2}_X$ in the same way as $U^{A_1B_1}_X$ using B_2 in place of B_1. Due to (D57), $U^{A_1B_1}_X$ in (D63) can be implemented by at most $N_{r_l}K_{r_l}$ level-$(l-1)$ CNOT-gate operations, and so can $U^{A_1B_2}_X$. Similarly, define $U^{A_1B_3B_4}_{X,\mathrm{C},3}$ in Fig. D.8 as the product of commuting operators $CU^{A_1B_3B_4}_{X,\mathrm{C},3,k^{(l)}}$ for all $k^{(l)} \in \{1,\dots,K^{(l)}\}$, i.e., under the mapping (D55),

$$\begin{aligned} &U^{A_1B_3B_4}_{X,\mathrm{C},3} \\ &\coloneqq \prod_{k^{(l)}=1}^{K^{(l)}} CU^{A_1B_3B_4}_{X,\mathrm{C},3,k^{(l)}} \\ &= \prod_{k=1}^{K_{r_l}} \prod_{n=1}^{N_{r_l}} \left(\prod_{k^{(l-1)}=1}^{K^{(l-1)}} \prod_{k^{(l-1)\prime}=1}^{K^{(l-1)}} \left(C^{A_1,k,k^{(l-1)}} P^{B_3,n,k^{(l-1)\prime}}_{X,3,3,k^{(l-1)},k,n,k^{(l-1)\prime}} \otimes C^{A_1,k,k^{(l-1)}} P^{B_4,n,k^{(l-1)\prime}}_{X,3,4,k^{(l-1)},k,n,k^{(l-1)\prime}} \right) \right), \end{aligned} \tag{D64}$$

and $U^{A_1B_3B_4}_{X,\mathrm{C},4}$ as the product of $CU^{A_1B_3B_4}_{X,\mathrm{C},4,k^{(l)}}$ for all $k^{(l)} \in \{1,\dots,K^{(l)}\}$. Note that in the last equality of (D64), we can reorder $\prod_{k^{(l-1)}=1}^{K^{(l-1)}} \prod_{n=1}^{N_{r_l}}$ into $\prod_{n=1}^{N_{r_l}} \prod_{k^{(l-1)}=1}^{K^{(l-1)}}$, since operators $P^{B_3,n,k^{(l-1)\prime}}_{X,3,3,k^{(l-1)},k,n,k^{(l-1)\prime}} \otimes P^{B_4,n,k^{(l-1)\prime}}_{X,3,4,k^{(l-1)},k,n,k^{(l-1)\prime}}$ with different n act on different level-$(l-1)$ registers of B_1 and B_2, and thus their controlled versions $C^{A_1,k,k^{(l-1)}} P^{B_3,n,k^{(l-1)\prime}}_{X,3,3,k^{(l-1)},k,n,k^{(l-1)\prime}} \otimes C^{A_1,k,k^{(l-1)}} P^{B_4,n,k^{(l-1)\prime}}_{X,3,4,k^{(l-1)},k,n,k^{(l-1)\prime}}$ for all $k^{(l-1)}$ and n commute with each other. Due to (D59), $U^{A_1B_3B_4}_{X,\mathrm{C},3}$ in (D64) can be implemented by at most $2N_{r_l}K_{r_l}$ level-$(l-1)$ two-register Clifford-gate abbreviations, and so can $U^{A_1B_3B_4}_{X,\mathrm{C},4}$.

In the following, we will count the number of one- and two-register gates in the part for implementing the Clifford unitary (D52), which has been deferred in the above explanations. We here prove that any Clifford unitary acting on the $2N_{r_l}K^{(l-1)}$ qubits in $2N_{r_l}$ level-$(l-1)$ registers, including (D52), can be implemented by a level-$(l-1)$ circuit composed of two-register Clifford gates with their number bounded by

$$O\left(N_{r_l}^2\right). \tag{D65}$$

Note that in a special case where each level-$(l-1)$ register reduces to only one qubit, Ref. [32] has shown an improved bound $O\left(N_{r_l}^2/\log(N_{r_l})\right)$ that may be better than (D65) by a logarithmic factor, but we leave as an open problem whether such an improvement is still possible in our more general case where each level-$(l-1)$ register may have more than one qubit. The depth of a stabilizer circuit could be shortened by parallelizing the circuit using auxiliary qubits [40, 41], but this further optimization is not considered here.

Our proof of (D65) uses the fact that any stabilizer circuit for implementing a Clifford unitary can be rewritten into an equivalent stabilizer circuit consisting of 11 rounds in a sequence, where each round is given by a cir-

cuit composed only of one type of Clifford gate, i.e., H, CNOT, S, CNOT, S, CNOT, H, S, CNOT, S, and CNOT, in this order [32]. This decomposition of the stabilizer circuit into one- and two-qubit gates can be performed feasibly using Gaussian elimination as shown in Ref. [32]. Note that Ref. [43] also provides a similar equivalence of stabilizer circuits, which may also be used here. Each round of H or S gates is performed with the level-$(l-1)$ two-register Clifford-gate abbreviation in Fig. D.1 acting on each of the $2N_{r_l}$ level-$(l-1)$ registers at most once. As for a round of CNOT gates, we here show that we can implement an arbitrary $2N_{r_l}K^{(l-1)}$-qubit CNOT circuit by a circuit composed only of $O(N_{r_l}^2)$ two-register Clifford gates on $2N_{r_l}$ level-$(l-1)$ registers.

To analyze the round of CNOT gates, as in Ref. [44], we represent the $2N_{r_l}K^{(l-1)}$-qubit CNOT circuit as a linear reversible transformation over the finite field $\mathbb{F}_2$, i.e., a $2N_{r_l}K^{(l-1)} \times 2N_{r_l}K^{(l-1)}$ binary matrix A, in such a way that the CNOT circuit linearly transforms the standard basis $\{|x\rangle\}$ of these $2N_{r_l}K^{(l-1)}$ qubits as

$$|x\rangle \mapsto |xA\rangle\,, \tag{D66}$$

where $x = (x_1, \ldots, x_{2N_{r_l}K^{(l-1)}}) \in \mathbb{F}_2^{2N_{r_l}K^{(l-1)}}$ is a row binary vector that is ordered in such a way that its ith element x_i with

$$i = (j-1)N_{r_l}K^{(l-1)} + (n-1)K^{(l-1)} + k^{(l-1)} \tag{D67}$$

represents the standard basis $\{|x_i\rangle : x_i \in \mathbb{F}_2 = \{0,1\}\}$ of the $k^{(l-1)}$th qubit in the nth level-$(l-1)$ register of B_{j+2}, for $k^{(l-1)} \in \{1, \ldots, K^{(l-1)}\}$, $n \in \{1, \ldots, N_{r_l}\}$, and $j \in \{1, 2\}$. We will decompose A into linear reversible transformations implementable by two-register Clifford gates, using Gaussian elimination.

Our decomposition of A representing the round of CNOT gates is as follows. We consider A to be a $2N_{r_l} \times 2N_{r_l}$ block matrix, where each $K^{(l-1)} \times K^{(l-1)}$ block represents the transformation involving the qubits in one or two level-$(l-1)$ registers. Any linear reversible transformation is implementable by CNOT gates as shown in Ref. [44], but our decomposition will be different from that in Ref. [44] in that we here use the two-register Clifford gates rather than single-qubit CNOT gates. Each two-register Clifford gate can apply an arbitrarily large number of CNOT gates between two level-$(l-1)$ registers at once, implementing an arbitrary linear reversible transformation of the basis of the qubits in the two level-$(l-1)$ registers. To show the decomposition, we use the lower-upper (LU) decomposition of A, i.e.,

$$A = PLU, \tag{D68}$$

where P is a permutation matrix, and L and U are lower and upper triangular matrices, respectively. The permutation matrix P has $2N_{r_l} \times 2N_{r_l}$ blocks and is implementable with $\binom{2N_{r_l}}{2}$ two-register Clifford gates, where each two-register Clifford gate performs SWAP gates between qubits in the two level-$(l-1)$ registers. The lower triangular matrix L can be written in the form of the block matrix as

$$L = \begin{pmatrix} L_{1,1} & 0 & 0 & \cdots & 0 \\ L_{2,1} & L_{2,2} & 0 & \cdots & 0 \\ L_{3,1} & L_{3,2} & L_{3,3} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ L_{2N_{r_l},1} & L_{2N_{r_l},2} & \cdots & L_{2N_{r_l},2N_{r_l}-1} & L_{2N_{r_l},2N_{r_l}} \end{pmatrix}, \tag{D69}$$

where each block $L_{n,n'}$ is a $K^{(l-1)} \times K^{(l-1)}$ matrix, and $L_{n,n}$ on the diagonal is a nonsingular lower triangular matrix since A is reversible. Then we decompose

$$L = DL', \tag{D70}$$

where D is a block diagonal matrix

$$D = \begin{pmatrix} L_{1,1} & 0 & 0 & \cdots & 0 \\ 0 & L_{2,2} & 0 & \cdots & 0 \\ 0 & 0 & L_{3,3} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & L_{2N_{r_l},2N_{r_l}} \end{pmatrix}, \tag{D71}$$

and L' is a block matrix with identity matrices on the diagonal

$$L' = \begin{pmatrix} \mathbb{1} & 0 & 0 & \cdots & 0 \\ L'_{2,1} & \mathbb{1} & 0 & \cdots & 0 \\ L'_{3,1} & L'_{3,2} & \mathbb{1} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ L'_{2N_{r_l},1} & L'_{2N_{r_l},2} & \cdots & L'_{2N_{r_l},2N_{r_l}-1} & \mathbb{1} \end{pmatrix}. \tag{D72}$$

To implement D, we use N_{r_l} two-register Clifford gates, where each two-register Clifford gate implements a linear reversible transformation acting nontrivially on two blocks on the diagonal. As for L', each off-diagonal block is implementable with one two-register Clifford gate; for example, we can decompose

$$\begin{pmatrix} \mathbb{1} & 0 & 0 & \cdots & 0 \\ L'_{2,1} & \mathbb{1} & 0 & \cdots & 0 \\ L'_{3,1} & L'_{3,2} & \mathbb{1} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ L'_{2N_{r_l},1} & L'_{2N_{r_l},2} & \cdots & L'_{2N_{r_l},2N_{r_l}-1} & \mathbb{1} \end{pmatrix} \tag{D73}$$

$$= \begin{pmatrix} \mathbb{1} & 0 & 0 & \cdots & 0 \\ L'_{2,1} & \mathbb{1} & 0 & \cdots & 0 \\ 0 & 0 & \mathbb{1} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & \mathbb{1} \end{pmatrix} \begin{pmatrix} \mathbb{1} & 0 & 0 & \cdots & 0 \\ 0 & \mathbb{1} & 0 & \cdots & 0 \\ L'_{3,1} & L'_{3,2} & \mathbb{1} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ L'_{2N_{r_l},1} & L'_{2N_{r_l},2} & \cdots & L'_{2N_{r_l},2N_{r_l}-1} & \mathbb{1} \end{pmatrix} \tag{D74}$$

$$= \begin{pmatrix} \mathbb{1} & 0 & 0 & \cdots & 0 \\ L'_{2,1} & \mathbb{1} & 0 & \cdots & 0 \\ 0 & 0 & \mathbb{1} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & \mathbb{1} \end{pmatrix} \begin{pmatrix} \mathbb{1} & 0 & 0 & \cdots & 0 \\ 0 & \mathbb{1} & 0 & \cdots & 0 \\ L'_{3,1} & 0 & \mathbb{1} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & \mathbb{1} \end{pmatrix} \begin{pmatrix} \mathbb{1} & 0 & 0 & \cdots & 0 \\ 0 & \mathbb{1} & 0 & \cdots & 0 \\ 0 & L'_{3,2} & \mathbb{1} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ L'_{2N_{r_l},1} & L'_{2N_{r_l},2} & \cdots & L'_{2N_{r_l},2N_{r_l}-1} & \mathbb{1} \end{pmatrix} = \cdots, \tag{D75}$$

where the linear reversible transformation represented by each matrix with only one off-diagonal block, such as $L'_{2,1}$ and $L'_{3,1}$ in the last line, is implementable with one two-register Clifford gate. By repeating this decomposition from left to right and from up to bottom as in the Gaussian elimination [44], we can implement all the ${}^{2N_{r_l}(2N_{r_l}-1)}/_{2}$ off-diagonal blocks of L' with ${}^{2N_{r_l}(2N_{r_l}-1)}/_{2}$ two-register Clifford gates. We can decompose U in the same way as L using the upper triangular matrices in place of the lower triangular matrices. As a result, the required number of two-register Clifford gates for implementing A is bounded by

$$\binom{2N_{r_l}}{2} + 2 \times \left(N_{r_l} + \frac{2N_{r_l}(2N_{r_l}-1)}{2} \right) = O(N_{r_l}^2). \tag{D76}$$

That is, the required number of two-register Clifford gates for implementing each round of CNOT gates is bounded by $O(N_{r_l}^2)$.

Therefore, the number of one- and two-register gates in the part for implementing the Clifford unitary (D52) is bounded by $O(N_{r_l}^2)$ as claimed in (D65), dominated by those for the rounds of CNOT gates. Since each two-register Clifford-gate abbreviation may have runtime at most $O(\log(N^{(l-1)})) = O(\log(N^{(l)}))$ as shown in (D36), the depth of the part for implementing the Clifford unitary (D52) is

$$O(N_{r_l}^2 \log(N^{(l)})). \tag{D77}$$

The part of $U_X^{A_1B_1}$, $U_X^{A_1B_2}$, $U_{X,\mathrm{C}}^{(3)A_1B_3B_4}$, and $U_{X,\mathrm{C}}^{(4)A_1B_3B_4}$ in the verification have $O(N_{r_l}K_{r_l}) = O(N_{r_l}^2)$ two-register Clifford gates as can be seen from (D63) and (D64). Due to the $O(\log(N^{(l)}))$ runtime of a two-

register Clifford-gate abbreviation, the required depth for implementing all these gates is also

$$O(N_{r_l}^2 \log(N^{(l)})). \tag{D78}$$

As for other non-dominant parts, due to (D49), each level-l initial-state preparation gadget has the $O(N_{r_l}\log(N_{r_l}))$ depth. Due to (D51), each level-l error-correction gadget has the $O(N_{r_l}\log(N_{r_l}))$ depth. The other parts are implementable within a constant depth. Finally, the depth of the second run of the gadget is smaller than that of the first run; thus, the depth of the gadget including the first and second runs is at most twice as long as the first run.

Consequently, the depth of the level-$(l-1)$ circuit in Fig. D.7 is

$$O(N_{r_l}^2 \log(N^{(l)})), \tag{D79}$$

dominated by (D77) and (D78). The required number of bits for classical computation is $O(N^{(l)2})$, dominated by those for storing the $O(N^{(l)}) \times O(N^{(l)})$ matrix.

As in (C20), the Clifford-state preparation operation can be invoked with an argument for representing $U_{\rm C}$, and below we discuss the on-demand function of the corresponding gadget. Note that, as a special case, $U_{\rm C}$ in the form of (C20) covers the implementation of the single-qubit HZ gates on a level-l register (while acting trivially on the other) used for the correction in the level-l $R(\pm\pi/4)$-gate abbreviation. The required function for the on-demand gadget is to implement a logical two-register Clifford unitary $U_{\rm C}$ acting on the two encoded level-l registers in the tensor-product form of (C20), i.e., $U_{\rm C} = \bigotimes_{k^{(l)}=1}^{K^{(l)}} U_{{\rm C},l}^{(k^{(l)})}$, where $U_{{\rm C},l}^{(k^{(l)})}$ is a two-qubit Clifford unitary acting on the $k^{(l)}$th qubit in each of the two level-l registers. In the on-demand case, several parts of the level-$(l-1)$ circuit of the gadget in Fig. D.7 change depending on $U_{\rm C}$. The parts that may change are the Clifford unitary (D52) for implementing logical $U_{\rm C}$ and the control unitaries $U_X^{A_1B_1}$, $U_X^{A_1B_2}$, $U_{X,{\rm C},3}^{A_1B_3B_4}$, and $U_{X,{\rm C},4}^{A_1B_3B_4}$ in (D63) and (D64) in the verification for measuring the logical X operators in (D53) (as well as those of Z in (D54)). In this case, all the level-$(l-1)$ two-register Clifford-gate abbreviations that depend on $U_{\rm C}$ in Fig. D.7 must be on-demand ones. In particular, the level-$(l-1)$ on-demand two-register Clifford-gate abbreviations are used for the Clifford unitary (D52) and the control unitaries $U_{X,{\rm C},3}^{A_1B_3B_4}$ and $U_{X,{\rm C},4}^{A_1B_3B_4}$ of (D64) (as well as those of Z), as shown in Fig. D.7. In the following, we show that these level-$(l-1)$ on-demand abbreviations are available.

To show the availability of the level-$(l-1)$ on-demand abbreviations in implementing logical $U_{\rm C}$ in the tensor-product form (C20), we here show that the unitaries of these level-$(l-1)$ on-demand abbreviations are always written in a tensor-product form

$$\bigotimes_{k^{(l-1)}=1}^{K^{(l-1)}} U_{{\rm C},l-1}^{(k^{(l-1)})}, \tag{D80}$$

as required at level $(l-1)$. To show this, under the mapping $k^{(l)} \mapsto (k, k^{(l-1)})$ of (B23), we write $U_{\rm C} = \bigotimes_{k^{(l)}=1}^{K^{(l)}} U_{{\rm C},l}^{(k^{(l)})}$ as

$$U_{\rm C} = \bigotimes_{k^{(l-1)}=1}^{K^{(l-1)}} \left(\bigotimes_{k=1}^{K_{r_l}} U_{{\rm C},l}^{(k,k^{(l-1)})} \right), \tag{D81}$$

where $U_{{\rm C},l}^{(k,k^{(l-1)})} \coloneqq U_{{\rm C},l}^{(k^{(l)})}$ acts on the kth logical qubits of the $k^{(l-1)}$th code blocks of $\mathcal{Q}_{r_l}$ for the two level-l registers. Moreover, the Clifford unitary $U_{\rm encode} = V^{\otimes K^{(l-1)}}$ for the encoding in (D47) is also a tensor product of the Clifford unitary V for the encoding of $\mathcal{Q}_{r_l}$. Thus, the Clifford unitary (D52) to implement logical $U_{\rm C}$ in the tensor-product form (C20) has a tensor-product form

$$\bigotimes_{k^{(l-1)}=1}^{K^{(l-1)}} \Bigg[(V\otimes V) \left(\left(\bigotimes_{k=1}^{K_{r_l}} U_{{\rm C},l}^{(k,k^{(l-1)})} \right) \otimes \mathbb{1}^{\otimes 2(N_{r_l}-K_{r_l})} \right) (V\otimes V)^\dagger \Bigg]. \tag{D82}$$

Then, by performing gate decomposition of each Clifford unitary

$$(V\otimes V) \left(\left(\bigotimes_{k=1}^{K_{r_l}} U_{{\rm C},l}^{(k,k^{(l-1)})} \right) \otimes \mathbb{1}^{\otimes 2(N_{r_l}-K_{r_l})} \right) (V\otimes V)^\dagger \tag{D83}$$

acting on the $k^{(l-1)}$th code blocks of $\mathcal{Q}_{r_l}$, we see that each two-register Clifford gate in the level-$(l-1)$ circuit also has the tensor-product form (D80) on level-$(l-1)$ registers, as required.

As for the availability of the level-$(l-1)$ on-demand abbreviations in implementing the control unitary $U_{X,{\rm C},3}^{A_1B_3B_4}$ of (D64) for the verification, we here show that the unitaries of these level-$(l-1)$ on-demand abbreviations are also written in a tensor-product form (D80). In the on-demand case, due to $U_{\rm C} = \bigotimes_{k^{(l)}=1}^{K^{(l)}} U_{{\rm C},l}^{(k^{(l)})}$ in (C20) as in the above analysis, the stabilizer operator $(U_{\rm C}(X_{3,k^{(l)}} \otimes \mathbb{1}^{\otimes(2K^{(l)}-1)})U_{\rm C}^\dagger)^{B_3B_4}$ in (D53) to be measured in the verification has the form of

$$(P_{3,k^{(l)}} \otimes P_{4,k^{(l)}}) \otimes \mathbb{1}^{\otimes(2K^{(l)}-2)}, \tag{D84}$$

where $P_{3,k^{(l)}}$ and $P_{4,k^{(l)}}$ are Pauli operators acting on the $k^{(l)}$th qubit in the level-l register of B_3 and B_4, respectively, and $\mathbb{1}^{\otimes(2K^{(l)}-2)}$ acts on the rest of the qubits. Then, $U_{X,{\rm C},3,k^{(l)}}^{B_3B_4}$ in (D58) for representing the logical operator in the form of (D84) reduces to

$$U_{X,{\rm C},3,k^{(l)}}^{B_3B_4} = \prod_{n=1}^{N_{r_l}} P_{X,3,3,k^{(l-1)},k,n,k^{(l-1)}}^{B_3,n,k^{(l-1)}} \otimes P_{X,3,4,k^{(l-1)},k,n,k^{(l-1)}}^{B_4,n,k^{(l-1)}}. \tag{D85}$$

Substituting $U^{B_3B_4}_{X,C,3,k^{(l)}}$ in (D58) with (D85) in the construction of $U^{A_1B_3B_4}_{X,C,3}$ in (D64), we see that each of the $2N_{r_l}K_{r_l}$ level-$(l-1)$ two-register Clifford gates for implementing $U^{A_1B_3B_4}_{X,C,3}$ is in the form of

$$\prod_{k^{(l-1)}=1}^{K^{(l-1)}} C^{A_1,k,k^{(l-1)}} P^{B_3,n,k^{(l-1)}}_{X,3,3,k^{(l-1)},k,n,k^{(l-1)}} = \bigotimes_{k^{(l-1)}=1}^{K^{(l-1)}} C^{A_1,k,k^{(l-1)}} P^{B_3,n,k^{(l-1)}}_{X,3,3,k^{(l-1)},k,n,k^{(l-1)}} \tag{D86}$$

or

$$\prod_{k^{(l-1)}=1}^{K^{(l-1)}} C^{A_1,k,k^{(l-1)}} P^{B_4,n,k^{(l-1)}}_{X,3,4,k^{(l-1)},k,n,k^{(l-1)}} = \bigotimes_{k^{(l-1)}=1}^{K^{(l-1)}} C^{A_1,k,k^{(l-1)}} P^{B_4,n,k^{(l-1)}}_{X,3,4,k^{(l-1)},k,n,k^{(l-1)}}, \tag{D87}$$

where $\prod$ becomes $\bigotimes$ since the controlled Pauli gates act on different qubits. Therefore, $U^{A_1B_3B_4}_{X,C,3}$ for the verification is composed of the two-register Clifford gates in the tensor-product form (D80) on level-$(l-1)$ registers, as required.

For this on-demand function, the runtime of classical computation for the gate decomposition is bounded as follows. To decompose the Clifford unitary into a stabilizer circuit in the rounds composed of one of the H, S, and CNOT gates, we use Gaussian elimination to convert the binary matrices into a canonical form as shown in Ref. [32]. As discussed with (C20), U_C in the tensor-product form is represented by $K^{(l)}$ 4×4 binary matrices. To obtain each $U^{(k^{(l-1)})}_{C,l-1}$ in (D80) from $U^{(k^{(l)})}_{C,l}$ for all $k^{(l)}$, we use the encoding unitary V for each code block $\mathcal{Q}_{r_l}$ rather than U_{encode} in (D47). By representing the Clifford unitary V as the $O(N_{r_l})\times O(N_{r_l})$ binary matrices, we perform the Gaussian elimination of the $O(N_{r_l})\times O(N_{r_l})$ matrices. As a whole, the Gaussian elimination of the $O(N_{r_l})\times O(N_{r_l})$ matrix requires $O(N_{r_l}^3)$ arithmetic operations, and using $O(N_{r_l}^2)$ parallel processes, we can run the Gaussian elimination within runtime $O(N_{r_l})$, which is dominant here. Therefore, for the part of preparing logical $(\mathbb{1}\otimes U_C)|\Phi^{(l)}\rangle$, we obtain the level-$(l-1)$ circuit composed of the rounds of the H, S, or CNOT gates within runtime $O(N_{r_l})$. As for the runtime of classical computation for the gate decomposition of the part for the verification, we obtain (D84) by calculating the conjugation of the single-qubit Pauli operator (and the single-qubit identity operator) by the two-qubit Clifford gate. The logical operator of (D84) is given by the tensor product of logical operators for at most one logical qubit in each of a pair of code blocks of $\mathcal{Q}_{r_l}$. Thus in the on-demand case, the controlled Pauli gates of (D86) and (D87) (as well as (D63)) for the verification can be obtained from logical operators of a pair of logical qubits of $\mathcal{Q}_{r_l}$ that appear in these controlled Pauli gates. The logical operator for each logical qubit of $\mathcal{Q}_{r_l}$ has been calculated during compilation, and in the same way, each pair of logical operators that may appear in these controlled Pauli gates in the on-demand case can be calculated and stored during compilation, which we here read from the classical memory during execution. As a whole, the runtime for the classical computation during the on-demand Clifford-state preparation gadget is

$$O(N_{r_l}), \tag{D88}$$

which is included in the time overhead in our setting via the insertion of the wait operations in Fig. D.7. Nevertheless, the runtime of this classical computation is smaller than the depth of the other dominant parts, i.e., (D77) and (D78). Thus, even for the on-demand Clifford-state preparation operation, the bound of the depth of the corresponding Clifford-state preparation gadget remains the same as (D79).

Magic-state preparation gadget: The level-l magic-state preparation gadget is constructed as in Fig. D.9. The gadget starts with a level-$(l-1)$ stabilizer circuit for encoding two sets of N_{r_l} level-$(l-1)$ registers into logical magic states. For each set, we use U_{encode} in (D47) to transform K_{r_l} level-$(l-1)$ registers prepared in $(R_y(\pi/4)|0\rangle)^{\otimes K^{(l-1)}}$ and $(N_{r_l}-K_{r_l})$ level-$(l-1)$ registers prepared in $|0\rangle^{\otimes K^{(l-1)}}$ into a logical state $(R_y(\pi/4)|0\rangle)^{\otimes K^{(l)}}$ as a whole, in the same way as the initial-state preparation gadget in Fig. D.5. As a result, we have logical $(R_y(\pi/4)|0\rangle)^{\otimes K^{(l)}}$ in each of the two sets. We call these two sets of N_{r_l} level-$(l-1)$ registers A and B, respectively, as shown in Fig. D.9.

Similar to the initial- and Clifford-state preparation gadgets, the encoding part is non-fault-tolerant, and hence, we need verification. In the verification, after ensuring that the state is in the code space of $\mathcal{Q}_{r_l}$, we measure all the logical stabilizer operators of logical $(R_y(\pi/4)|0\rangle)^{\otimes K^{(l)}}$ to detect logical errors with post-selection. Each state $R_y(\pi/4)|0\rangle$ is stabilized by

$$H(R_y(\pi/4)|0\rangle) = R_y(\pi/4)|0\rangle\,. \tag{D89}$$

To specify the logical state of these two encoded registers to be $(R_y(\pi/4)|0\rangle)^{\otimes K_{r_l}}\otimes(R_y(\pi/4)|0\rangle)^{\otimes K_{r_l}}$, we use a set of the $2K_{r_l}$ stabilizer operators of $(R_y(\pi/4)|0\rangle)^{\otimes K_{r_l}}\otimes(R_y(\pi/4)|0\rangle)^{\otimes K_{r_l}}$ given by

$$\Big\{(H^{\otimes k}\otimes\mathbb{1}^{\otimes(K_{r_l}-k)})\otimes(\mathbb{1}^{\otimes k}\otimes H^{\otimes(K_{r_l}-k)}),\ (\mathbb{1}^{\otimes k}\otimes H^{\otimes(K_{r_l}-k)})\otimes(H^{\otimes k}\otimes\mathbb{1}^{\otimes(K_{r_l}-k)})\Big\}_{k\in\{1,\dots,K_{r_l}\}}, \tag{D90}$$

where $\mathbb{1}^{\otimes 0}=1$ and $H^{\otimes 0}=1$. To measure the logical operators of these stabilizer operators in a transversal manner, we here use the fact that the logical operator $H^{\otimes K_{r_l}}$

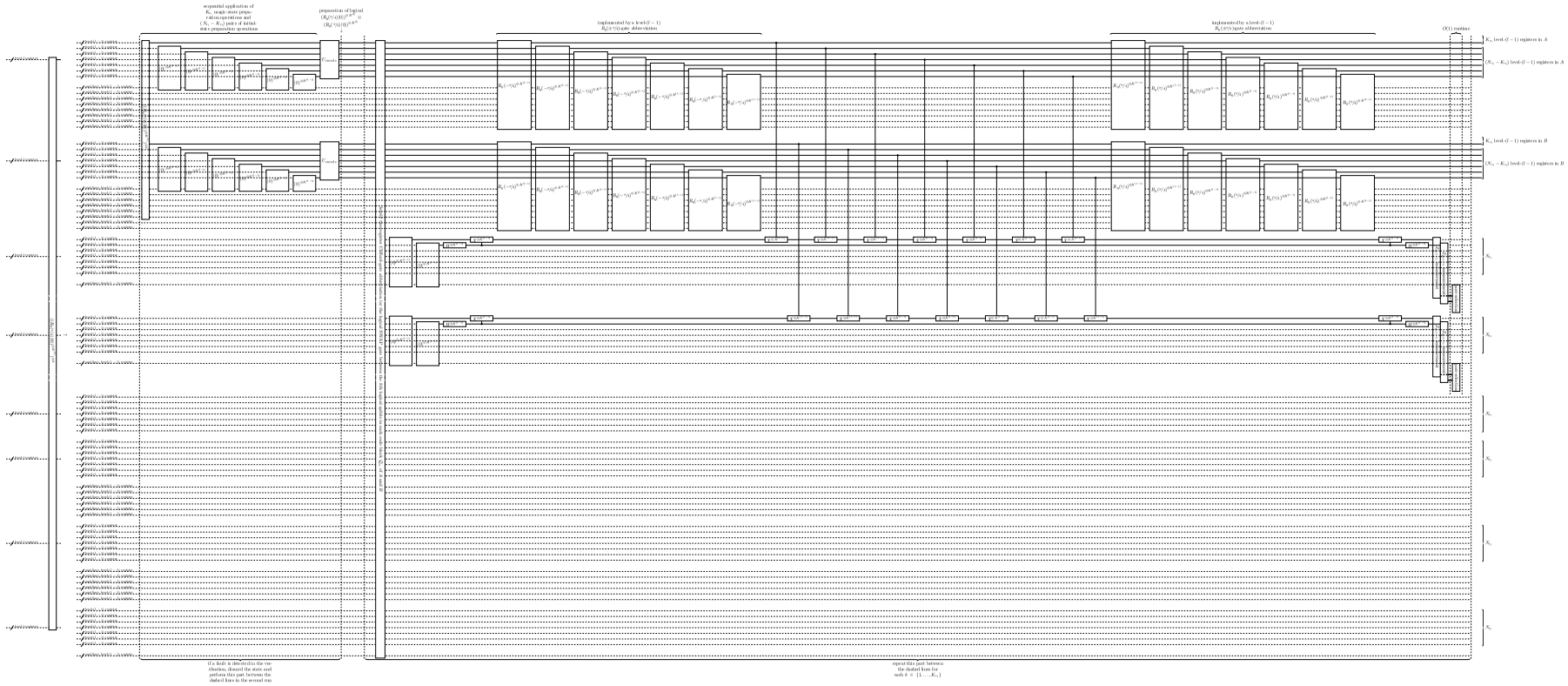


FIG. D.9. The level-l magic-state preparation gadget for preparing a logical state $((R_y(\pi/4)\,|0\rangle)^{\otimes K^{(l)}}) \otimes ((R_y(\pi/4)\,|0\rangle)^{\otimes K^{(l)}})$ of two encoded level-l registers, where the two sets of N_{r_l} level-$(l-1)$ registers for these two encoded level-l registers are denoted respectively by A and B. The white box for the level-l two-register Clifford-gate abbreviation is to be replaced with the corresponding level-$(l-1)$ circuit obtained from the level-l circuit in Fig. D.1 by replacing each level-l operation with the corresponding level-l gadget. The gadget starts with a level-$(l-1)$ stabilizer circuit for encoding, i.e., for transforming K_{r_l} level-$(l-1)$ registers in $(R_y(\pi/4)\,|0\rangle)^{\otimes K^{(l-1)}}$ and $(N_{r_l}-K_{r_l})$ level-$(l-1)$ registers in $|0\rangle^{\otimes K^{(l-1)}}$ into the logical state $(R_y(\pi/4)\,|0\rangle)^{\otimes K^{(l)}}$ for each of A and B, in the same way as the initial-state preparation gadget in Fig. D.5. In the verification, after ensuring that the state is in the code space of $\mathcal{Q}_{r_l}$, we measure the logical stabilizer operators (D90) of logical $(R_y(\pi/4)\,|0\rangle)^{\otimes K_{r_l}} \otimes (R_y(\pi/4)\,|0\rangle)^{\otimes K_{r_l}}$ for each pair of code blocks of $\mathcal{Q}_{r_l}$ in A and B, where the logical stabilizer operators are given according to $H(R_y(\pi/4)\,|0\rangle) = R_y(\pi/4)\,|0\rangle$. To measure each of the logical stabilizer operators iteratively, for $k = 1, \ldots, K_{r_l}$, the level-l two-register Clifford-gate abbreviation is used for implementing the logical SWAP gate between the kth logical qubits in each code block $\mathcal{Q}_{r_l}$ of A and B, and then logical $H^{\otimes K_{r_l}}$ is measured for each code block $\mathcal{Q}_{r_l}$ of A and B. For this measurement in each iteration, we extract $H = R_y(\pi/4)ZR_y(-\pi/4)$ to a level-$(l-1)$ register by combining the CNOT-gate operations with $R_y(\pm\pi/4)$ gates implemented by level-$(l-1)$ $R_y(\pm\pi/4)$-gate abbreviations in Fig. D.2. In each extraction, another level-$(l-1)$ register is used as flag qubits to make the verification fault-tolerant. If no error is detected from logical stabilizer operators (D90) and the flag qubits in these iterations, then the gadget outputs the logical $(R_y(\pi/4)\,|0\rangle)^{\otimes K_{r_l}} \otimes (R_y(\pi/4)\,|0\rangle)^{\otimes K_{r_l}}$ prepared in this first run; otherwise, this prepared state is discarded, and the gadget again performs the part between the dashed lines as shown in the figure and outputs the logical $(R_y(\pi/4)\,|0\rangle)^{\otimes K_{r_l}} \otimes (R_y(\pi/4)\,|0\rangle)^{\otimes K_{r_l}}$ prepared in this second run without verification.

acting on all the logical qubits of each code block $\mathcal{Q}_{r_l}$ is given by $H^{\otimes N_{r_l}}$ acting on all the qubits in the level-$(l-1)$ registers. To measure different operators in (D90), for $k = 1, \ldots, K_{r_l}$, we iteratively perform a logical SWAP gate between the kth logical qubit of each code block $\mathcal{Q}_{r_l}$ in A and B, and then measure logical $H^{\otimes K_{r_l}}$ of each code block $\mathcal{Q}_{r_l}$. After finishing all the K_{r_l} iterations, it is to be verified that all the logical qubits in A and B should be in a logical state stabilized by (D90), i.e., in logical $(R_y(\pi/4)\,|0\rangle)^{\otimes K^{(l)}} \otimes (R_y(\pi/4)\,|0\rangle)^{\otimes K^{(l)}}$.

More specifically, for each $k \in \{1, \ldots, K_{r_l}\}$, the circuit in Fig. D.9 iteratively performs a logical SWAP gate between a pair of the kth logical qubits in each code block $\mathcal{Q}_{r_l}$ of A and B, so that the first k logical qubits of $\mathcal{Q}_{r_l}$ that were initially in B are brought to A after the kth logical SWAP gate. Each of the logical SWAP gates is implemented by the level-l two-register Clifford-gate abbreviation in Fig. D.1. The first SWAP gate also serves the purpose of ensuring that the state is in the code space of $\mathcal{Q}_{r_l}$. In each iteration, the circuit in Fig. D.9 performs a measurement of logical $H^{\otimes K_{r_l}}$ of each code block $\mathcal{Q}_{r_l}$ in A and B. This measurement is assisted by two auxiliary level-$(l-1)$ registers prepared in $|0\rangle^{\otimes K^{(l-1)}}$, where one is used for extracting the measurement outcomes, and the other is used as the flag qubits [42]. This measurement can be performed via controlled H gates on the target in A and B controlled by the auxiliary level-$(l-1)$ register for the extraction, as in Refs. [8, 45, 46]. By decomposing $H = R_y(\pi/4)ZR_y(-\pi/4)$, we implement these controlled H gates using $R_y(\pm\pi/4)$ gates in Fig. D.2 and also using the X basis as the basis of the control qubits, which leads to the $R_y(\pm\pi/4)$ and CNOT part of the circuit in Fig. D.9. For each $k^{(l-1)} \in \{1, \ldots, K^{(l-1)}\}$, the outcome of measuring the operator $H^{\otimes N_{r_l}}$ on the N_{r_l} qubits forming the $k^{(l-1)}$th code block $\mathcal{Q}_{r_l}$ is extracted as that on the $k^{(l-1)}$th qubit in the first auxiliary level-$(l-1)$ register. Given that the measured state is in the code space, this outcome coincides with that of measuring logical $H^{\otimes K_{r_l}}$ in each code block $\mathcal{Q}_{r_l}$. In addition, an error on this auxiliary level-$(l-1)$ register that may propagate and cause an uncorrectable error is detected as a bit flip of the measurement outcomes on the flag qubits. Consequently, after the K_{r_l} iterations without detecting any error, it is to be verified that A and B should be in logical $(R_y(\pi/4)\,|0\rangle)^{\otimes K^{(l)}} \otimes (R_y(\pi/4)\,|0\rangle)^{\otimes K^{(l)}}$ stabilized by (D90), as long as the state of A and B is in the code space. As in the Clifford-state preparation gadget, the flip of each of these $O(N^{(l)})$-bit measurement outcomes can be checked using $O(N^{(l)})$ parallel processes immediately, i.e., in runtime $O(1)$.

If no error is detected either from the logical stabilizer operators (D90) and the flag qubits in these iterations, then the gadget outputs the logical $(R_y(\pi/4)\,|0\rangle)^{\otimes K^{(l)}} \otimes (R_y(\pi/4)\,|0\rangle)^{\otimes K^{(l)}}$ prepared in this first run. Otherwise, the gadget discards the state prepared in the first run and performs the same non-fault-tolerant level-$(l-1)$ stabilizer circuit for the preparation and outputs the logical $(R_y(\pi/4)\,|0\rangle)^{\otimes K^{(l)}} \otimes (R_y(\pi/4)\,|0\rangle)^{\otimes K^{(l)}}$ prepared in the second run without verification.

The magic-state preparation gadget is fault-tolerant, that is, satisfies the conditions (D15) and (D16). The gadget can have only the first run (with the success in the verification) or both the first and second runs. but in the same way as the analysis of the fault tolerance of the Clifford-state preparation gadgets, we here explain the cases where the gadget has only the first run with $s = 1$. In particular, the analysis reduces to the following three cases: (i) a case where a fault occurs on one of the level-$(l-1)$ locations on A in the part of the level-$(l-1)$ circuit in Fig. D.9 for preparing logical $(R_y(\pi/4)\,|0\rangle)^{\otimes K^{(l)}}$, (ii) a case where a fault occurs on one of the level-$(l-1)$ locations on B in the part for preparing logical $(R_y(\pi/4)\,|0\rangle)^{\otimes K^{(l)}}$, and (iii) a case where a fault occurs on one of the level-$(l-1)$ locations in the part of the level-$(l-1)$ circuit in Fig. D.9 for the verification. In case (i), since the single fault occurs at a location on A, the register B is prepared in logical $(R_y(\pi/4)\,|0\rangle)^{\otimes K^{(l)}}$ with no error after the preparation part. In addition, the level-l two-register Clifford-gate abbreviation in Fig. D.1 for the logical SWAP gate ensures that the state in A and B is in the code space of $\mathcal{Q}_{r_l}$ since this Clifford-gate part has no fault. Then the measurements of the $2K_{r_l}$ logical stabilizer operators (D90) in each of the $K^{(l-1)}$ code space $\mathcal{Q}_{r_l}$ are faultlessly carried out. Conditioned on the success of post-selection, it is guaranteed that the logical state of all the $2K_{r_l} \times K^{(l-1)} = K^{(l)}$ logical qubits should be $(R_y(\pi/4)\,|0\rangle)^{\otimes K^{(l)}} \otimes (R_y(\pi/4)\,|0\rangle)^{\otimes K^{(l)}}$. In case (ii), the fault tolerance can be shown in the same way as case (i) with reversal of A and B. As for case (iii), each of A and B before the verification is in the logical state $(R_y(\pi/4)\,|0\rangle)^{\otimes K^{(l)}}$ without any error, and the level-$(l-1)$ error in the verification does not cause uncorrectable errors at the end of the level-$(l-1)$ circuit. In particular, a fault in implementing the level-l two-register Clifford-gate abbreviation in Fig. D.1 does not cause uncorrectable errors. To see this, if we replace level-l elementary operations in the abbreviation of Fig. D.1 with the corresponding level-l gadgets and assume that these gadgets satisfy the fault-tolerant properties from (D10) to (D20), then we can confirm that the resulting level-$(l-1)$ circuit for the abbreviation implementing the logical two-register gate satisfies (D19) and (D20). Moreover, conditioned on the success of the post-selection, a fault among level-$(l-1)$ locations acting on auxiliary level-$(l-1)$ registers in the verification does not propagate and cause logical errors since such error propagation can be detected due to the technique of flag qubits [42], as in protocols in Refs. [8, 45, 46] for magic state preparation with this technique. A correctable error before the faultless two-register Clifford-gate abbreviation is corrected in the same way as level-l error-correction gadgets in Fig. D.6 correcting such an error. Finally, a correctable

error before the faultless measurement of the logical stabilizer operators is kept correctable in the same way as the Clifford-state preparation gadget. These cases show the fault tolerance.

The depth of the level-$(l-1)$ circuit for the level-l magic-state preparation gadget is dominated by the level-l two-register Clifford-gate abbreviations for the logical SWAP gates in the verification. The verification part iterates the level-l two-register Clifford-gate abbreviation and the measurements of the logical stabilizer operators K_{r_l} times in total. The part corresponding to the level-l two-register Clifford-gate abbreviation in Fig. D.1 has the depth of $O(N_{r_l}^2 \log(N^{(l)}))$, dominated by the level-l Clifford-state preparation gadget as shown in (D79). As for the other parts in the verification, each level-$(l-1)$ $R_y(\pm\pi/4)$-gate abbreviation has the depth of $O(\log(N^{(l)}))$ as shown in (D37), and in each iteration, the level-$(l-1)$ $R_y(\pm\pi/4)$-gate abbreviations are used sequentially $O(N_{r_l})$ times; that is, the depth of the level-$(l-1)$ $R_y(\pm\pi/4)$-gate abbreviations in each iteration is $O(N_{r_l} \log(N^{(l)}))$. The depth of performing the CNOT-gate operations for measuring the logical stabilizer operators is $O(N_{r_l})$ in each iteration. The other parts in each iteration have a constant depth including the $O(1)$ runtime of classical computation for the post-selection. Repeating K_{r_l} times, the verification part as a whole has the depth of $K_{r_l} \times O(N_{r_l}^2 \log(N^{(l)}))$, which dominate the depth of the gadget. As for other non-dominant parts, the depth of the part for implementing U_{encode} is $O(N_{r_l} \log(N_{r_l}))$ as shown in (D49). The preparation operations at the beginning of the gadget are performed sequentially in the $O(N_{r_l})$ depth. Finally, the depth of the second run of the gadget is smaller than that of the first run; thus, the depth of the gadget including the first and second runs is at most twice as long as the first run. Consequently, the depth of the gadget is

$$K_{r_l} \times O(N_{r_l}^2 \log(N^{(l)})) = O(N_{r_l}^3 \log(N^{(l)})). \tag{D91}$$

The required number of bits for the classical computation is dominated by $O(N^{(l)2})$ bits used for the level-l Clifford-state preparation gadgets included in this gadget.

Appendix E: Proof of existence of threshold for doubly exponential error suppression

In this section, we present the proof of the existence of a threshold for doubly exponential error suppression in the concatenation level L achieved by the fault-tolerant protocol constructed in Secs. C and D. To prove the existence of a threshold, the existing fault-tolerant protocol with constant space overhead in Refs. [3, 6, 7] assumes that classical computation for the decoder for the quantum LDPC codes runs instantaneously in zero time, and it has been unknown whether constant-space-overhead FTQC is possible if the classical computation has nonzero runtime that grows on the large scales, as in our setting. The problem of requiring the zero (i.e., non-growing) runtime of classical computation is rooted even in conventional protocols based on vanishing-rate quantum LDPC codes such as the 2D surface code, where decoders must process the stream of syndrome data at the rate it is received even if the code distance grows [47–51]. After all, if we need to wait for a growing runtime of the decoder for the large-size quantum LDPC codes for sufficient error suppression, physical qubits suffer from more errors during waiting; in this case, the assumption of having a constant physical error rate between performing the error corrections may be violated, and hence, the proof technique for the threshold theorem based on the quantum LDPC codes is no longer applicable in a straightforward way. For example, in the local stochastic error model as considered in the analysis of the existing constant-space-overhead protocol [3] and our work, time-correlated errors at physical error rate p can occur in such a way that each time period of time length $1/p$ on each physical qubit always includes exactly one faulty location, uniformly at random over the $1/p$ locations. Then, when the runtime T_{decoder} of the decoder on a large scale exceeds a constant time

$$T_{\text{decoder}} > 1/p, \tag{E1}$$

physical qubits simultaneously suffer from errors while waiting for the decoder with unit probability, which are uncorrectable. To correct a general class of errors in such a situation, a novel technique should be needed, such as another nontrivial time scheduling of quantum error correction and gate implementation, progressing beyond those for the existing constant-space-overhead protocol based on quantum LDPC codes [3, 6, 7]. By contrast, the crucial contribution here is to develop the technique of using the concatenated code with the non-vanishing rate, rather than the quantum LDPC codes. Owing to this technique, the assumption that the runtime of classical computation should never grow on large scales can be avoided. Our analysis proves the existence of a threshold in the local stochastic error model even if we take into account the growth of the nonzero runtime of classical computation.

Note that, for the quantum expander code used for the existing constant-space-overhead protocol [3, 6, 7], Ref. [52] claims to construct a decoder that has $O(1)$ runtime on arbitrary large scales, which would avoid the above growth of the runtime of the decoder if implemented without any architectural time overhead. However, the $O(1)$ runtime of this decoder depends on an assumption that the decoder can receive an arbitrarily long bit string as the input (e.g., by reading it from a classical memory at the hardware level) and can return an arbitrarily long bit string as the output (e.g., by writing it into the memory) within only a constant time on the arbitrarily large scales. In particular, the decoder receives measurement syndromes for the large-size code block of the quantum LDPC code as the input, runs many parallel processes in such a way that

each process runs in $O(1)$ time, and then, from the results of these processes, returns estimation of the overall errors in the large-size code block as the output. If such an input/output model of the decoder were implemented naively, to meet the requirement of the constant time in Ref. [52], the input/output speed between the decoder and the memory would diverge to infinity on large scales. In practice, hardware implementations of decoders in physical experiments are challenged by architectural overheads [53, 54], and at the hardware level, classical input/output interface may incur a growing architectural time overhead on large scales. It is unknown how one can implement the input/output model of the above decoder within a constant time at the hardware level. Problematically, it has been difficult for the existing constant-space-overhead protocol [3, 6, 7] to tolerate such growing architectural time overheads in implementing the decoder due to (E1). By contrast, the following analysis of the existence of a threshold shows that our fault-tolerant protocol can tolerate polynomially growing architectural time overheads, as discussed later in (E19).

In particular, by suitably modifying the proof of the threshold theorem for the concatenated code [1], our proof is done by counting the number of locations in extended rectangles (ExRecs), so as to bound a probability of faults at a concatenation level in terms of that at a lower level. Given a level-l circuit and the corresponding level-$(l-1)$ circuit, for each level-l location of the level-l circuit, the level-l ExRec corresponding to the level-l location is defined as a part of the level-$(l-1)$ circuit consisting of the level-l gadget corresponding to the level-l location, all the level-l error-correction gadgets placed between the location and the adjacent locations, and the level-$(l-1)$ wait operations inserted between these gadgets. A difference from Ref. [1] arises from the fact that the error correction on each set of $O(N_{r_l})$ encoded level-l registers in our fault-tolerant protocol are performed in a synchronized way as shown in Fig. C.2, and for the synchronization, our protocol inserts level-$(l-1)$ wait operations between the gadgets. Due to this difference, a level-l ExRec for a level-l location in our protocol includes the level-$(l-1)$ wait operations between the level-l gadget for the location and each adjacent level-l error-correction gadget. See Fig. E.1 for an illustration of ExRecs in our protocol.

Let $A(l)$ be the maximum number of pairs of locations in the level-$(l-1)$ circuit of a level-l ExRec, where the maximum is taken over all the possible choices of the level-l ExRecs. In the following, we discuss an upper bound of $A(l)$ in the cases of $l < L$ while the case of $l = L$ will also be discussed subsequently. To bound $A(l)$, it is crucial to use the fact that the maximum depth $G(l)$ of the level-$(l-1)$ circuit for any level-l gadget used in our fault-tolerant protocol is bounded polynomially in N_{r_l} by (D23) and (D24), i.e.,

$$G(l) = O(\mathrm{poly}(N_{r_l})) = \exp(O(l)), \tag{E2}$$

which has been checked in Sec. D and is dominated by (D91) of the magic-state preparation gadgets in the worst case. Note that $G(l)$ is an upper bound of the sum of the depth of the gadget and that of wait operations inserted for synchronizing error correction as shown in Fig. C.2. We stress that $G(l)$ in our analysis of Secs. C and D includes the wait operations to wait for nonzero-time classical computation such as those in the decoder and the gate teleportation. A level-l elementary operation acts on at most 8 level-l registers, where 8 are used for the level-l magic-state preparation in (C8). A level-l gadget corresponding to a level-l elementary operation uses $(N_{r_l}+8)$ level-$(l-1)$ registers per encoded level-l register. Thus, the number of level-$(l-1)$ locations of the level-l gadget and the wait operations inserted for synchronizing error correction is at most (see also Fig. C.2)

$$8(N_{r_l}+8)G(l). \tag{E3}$$

As for error correction, a level-l error-correction gadget uses at most $(3N_{r_l}+2)$ level-$(l-1)$ registers as shown in (C14). Thus, the number of level-$(l-1)$ locations used for each level-l error-correction gadget is

$$(3N_{r_l}+2)G(l). \tag{E4}$$

As shown in Fig. C.2, each time period for error correction performs at most $(N_{r_{l+1}}+8)$ error-correction gadgets sequentially. Each level-l elementary operation has at most 4 adjacent level-l elementary operations, as in the case of (C3), (C4), and (C7); that is, each level-l ExRec has at most 4 level-l error-correction gadgets in addition to one level-l gadget to implement a level-l elementary operation. As a whole, the number of level-$(l-1)$ locations in a level-l ExRec is upper bounded by (see also Fig. C.2)

$$\begin{aligned} &8(N_{r_l}+8)G(l) + 4(N_{r_{l+1}}+8)(3N_{r_l}+2)G(l) \\ &= [8(N_{r_l}+8) + 4(N_{r_{l+1}}+8)(3N_{r_l}+2)]G(l). \end{aligned} \tag{E5}$$

Therefore, in the cases of $l < L$, it holds that

$$A(l) \leqq \binom{[8(N_{r_l}+8) + 4(N_{r_{l+1}}+8)(3N_{r_l}+2)]G(l)}{2} \tag{E6}$$

$$= O(\mathrm{poly}(N_{r_l})) \tag{E7}$$

$$= \exp(O(l)). \tag{E8}$$

As for the case of $l = L$, since the number of error-correction gadgets sequentially used in each time period of error correction is at most 9 ($\leqq N_{r_{L+1}}+8$) as shown in Fig. C.2, the above upper bound of $A(l)$ holds for $l = L$. Thus, to simplify the presentation, our analysis will use a constant factor $\alpha > 0$ such that

$$A(l) \leqq 2^{\alpha l} \tag{E9}$$

for all $l \in \{1, \ldots, L\}$. Note that these upper bounds may not be tight but are sufficient for the following analysis.

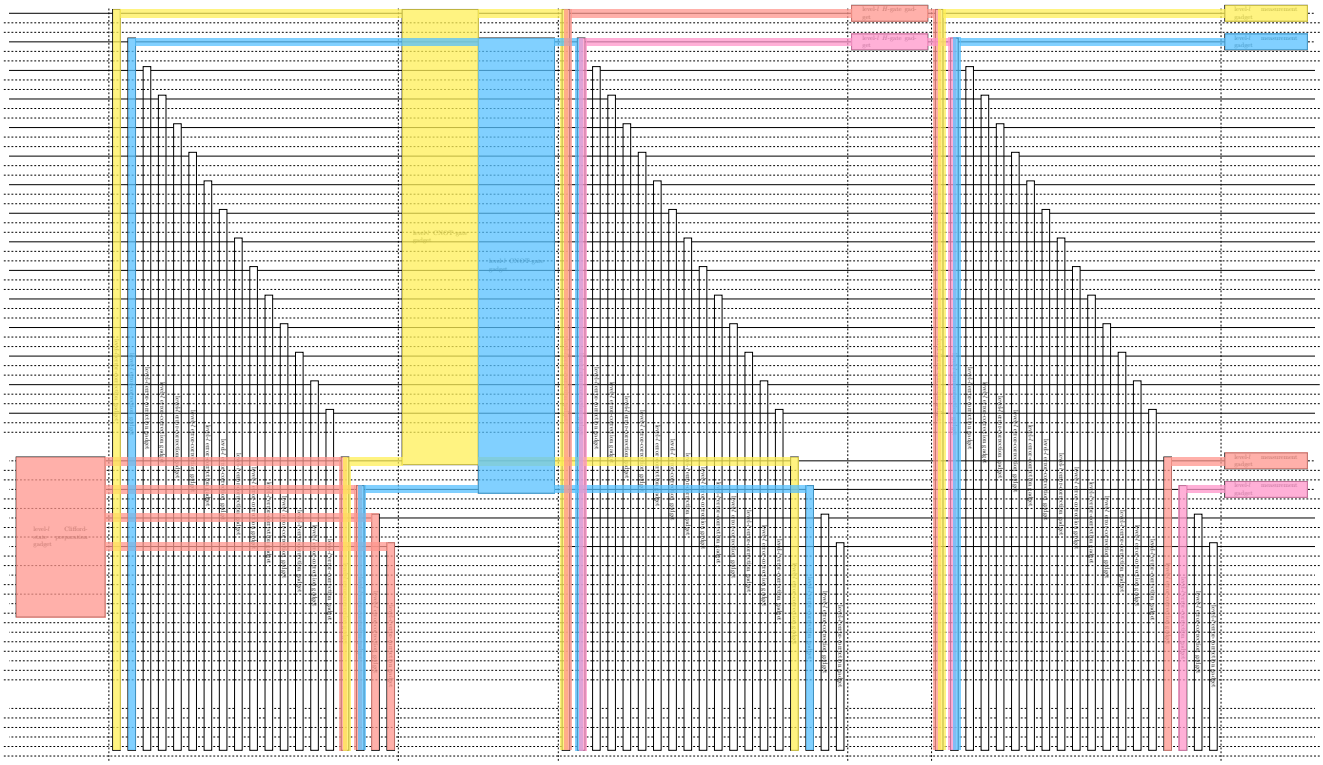


FIG. E.1. Extended rectangles (ExRecs) in the level-$(l-1)$ circuit on the right of Fig. C.2 for implementing a logical level-l circuit. Each level-l ExRec corresponding to a level-l location is a part of the level-$(l-1)$ circuit (colored in the figure) consisting of the level-l gadget corresponding to a level-l location, all the level-l error-correction gadgets placed between the location and the adjacent locations, and the level-$(l-1)$ wait operations inserted between these gadgets. Note that the figure does not color the level-l ExRecs corresponding to level-l wait operations for simplicity.

To bound the error rate of a level-l circuit in terms of those in the corresponding level-$(l-1)$ circuit, a conventional argument in Ref. [1] under the assumption of the local stochastic error model is applied here. In the same way as the local stochastic error model at the physical level as in (A5), for a level-l circuit at each level $l \in \{1, \dots, L\}$, we say that the circuit undergoes a local statistical error model if the faults occurring in the circuit satisfy the following: (i) a set S of faulty level-l locations in the level-l circuit is chosen with probability $p^{(l)}(S)$, and errors occur at the level-l locations in S in such a way that the level-l elementary operations at the level-l locations in S are replaced with an arbitrary quantum channel $\mathcal{E}$ that is consistent with the causal order of the locations in S; (ii) each level-l location i in the circuit has a parameter $p_{l,i}$ such that for any set R of level-l locations, the probability $\Pr\{S \supseteq R\}$ of having faults at every level-l location in R is at most

$$\prod_{i \in R} p_{l,i}. \tag{E10}$$

We call the parameter $p_{l,i}$ a logical error rate for a level-l location i in the level-l circuit. For simplicity of presentation, we consider a single parameter p_l such that $p_{l,i} \leqq p_l$ holds for all i. If obvious from the context, we may simply call p_0 the physical error rate, and p_l the logical error rate at level l.

For each l, suppose that level-l gadgets are fault-tolerant in the sense of the definitions of the equivalence relations shown in Sec. C, and that the level-$(l-1)$ circuit undergoes a local stochastic error model. Then, the argument in Ref. [1] proves that the level-l circuit also undergoes a local stochastic error model. As discussed in Sec. A with (A5), the level-0 circuit is assumed to undergo the local stochastic error model in our setting. Thus, using such an argument recursively, the level-l circuits at every level $l \in \{1, \dots, L\}$ are shown to undergo a local stochastic error model. Moreover, since no single fault at a level-$(l-1)$ location in a level-l ExRec leads to a fault of the corresponding level-l location, the argument in Ref. [1] shows that the logical error rate p_l at level l is bounded in terms of p_{l-1} at level $(l-1)$ by

$$p_l \leqq A(l)(p_{l-1})^2. \tag{E11}$$

Therefore, it follows from (E9) that

$$p_l \leqq 2^{\alpha l}(p_{l-1})^2; \tag{E12}$$

that is, p_l satisfies

$$2^{g(l)} p_l \leqq (2^{g(l-1)} p_{l-1})^2, \tag{E13}$$

where

$$g(l) \coloneqq \alpha(l+2). \tag{E14}$$

Using (E13) recursively, it holds that

$$p_L \leqq 2^{-g(L)}(2^{g(0)}p_0)^{2^L} \tag{E15}$$

$$= \frac{(2^{2\alpha}p_0)^{2^L}}{2^{\alpha(L+2)}}. \tag{E16}$$

Consequently, there exists a threshold

$$p_{\rm th} \geqq 2^{-2\alpha}(>0) \tag{E17}$$

such that for any nonzero physical error rate $p_0 \in (0, p_{\rm th})$ below the threshold, the logical error rate p_L decreases doubly exponentially as L increases, i.e.,

$$p_L \leqq \left(\frac{p_0}{p_{\rm th}}\right)^{2^L} p_{\rm th}, \tag{E18}$$

achieving the same scaling as the doubly exponential error suppression in L with the conventional concatenated codes [1]. This conclusion is summarized as Proposition 1 in the main text.

From the argument here, 2α on the right-hand side of (E17) is crucial for obtaining a concrete value of a lower bound of $p_{\rm th}$ while the exact evaluation of $p_{\rm th}$ may require numerical simulation. To make this lower bound of $p_{\rm th}$ larger, α should be small, and hence, it is essential to make the size of ExRecs (or $A(l)$ in (E9)) as small as possible. For further optimization of the ExRecs, it would be crucial to reduce the size of Clifford- and magic-state preparation gadgets and also optimize the scheduling of error correction rather than just performing it sequentially between level-l operations (as in Fig. C.2).

Note that the above proof of the doubly exponential error suppression in L holds even in more general cases of

$$A(l) = \exp(O(\mathrm{poly}(l))). \tag{E19}$$

For example, suppose that classical computation for the decoder or the gate teleportation in a gadget had a polynomial runtime $O(N^{(l)}) = \exp(O(l^2))$ in the code size $N^{(l)}$ of $\mathcal{Q}^{(l)}$, which would be much slower than our protocol but may be the case where the number of parallel processes in classical computation per level-l register is limited. Even in such a case, we would have $A(l) = \exp(O(l^2)) \leqq 2^{\alpha l^2}$ for some α, and with a choice of $g(l) = \alpha(l^2 + 4l + 6)$ in place of g in (E14), we can prove $p_L \leqq (2^{6\alpha}p_0)^{2^L}/2^{\alpha(L^2+4L+6)}$ from the same argument as the above. Then, a threshold $p_{\rm th} \geqq 2^{-6\alpha}(>0)$ in place of (E17) still exists. In summary, the above argument leads to the following theorem.

Theorem 2 (Threshold theorem and logical error rate)**.** Suppose that we have a fault-tolerant protocol using concatenation of $[[n_l, k_l, 2t+1]]$ quantum codes with $n_l = \exp(O(\mathrm{poly}(l)))$ at each concatenation level $l \in \{1, 2, \ldots\}$, where $t \geqq 1$, the level-l gadgets are fault-tolerant in terms of the equivalence relations translated from Ref. [1] as those in Sec. D in the case of $t = 1$, and each level-l ExRec is composed of $\exp(O(\mathrm{poly}(l)))$ level-$(l-1)$ elementary operations. Then, under the local stochastic error model at physical error rate $p_0 > 0$, there exists a nonzero threshold $p_{\rm th} > 0$ such that the level-l circuit for each l undergoes the local stochastic error model, and the logical error rate p_l at concatenation level l is bounded by

$$p_l \leqq \left(\frac{p_0}{p_{\rm th}}\right)^{(t+1)^l} p_{\rm th}. \tag{E20}$$

Remarkably, a practical threshold better than $p_{\rm th}$ in (E17) is achievable with minor modifications of our fault-tolerant protocol. Our results make it possible to achieve such a practical threshold with constant space overhead and parallelizability. Below we show techniques for such improvement of the threshold.

Firstly, instead of starting concatenation as $\mathcal{Q}_{r_1}, \mathcal{Q}_{r_2}, \ldots$, the 7-qubit code $\mathcal{Q}_3$ can be used in the first *constant* times of the concatenation; in this case, the sequence of codes in the concatenation becomes $\mathcal{Q}_3, \ldots, \mathcal{Q}_3, \mathcal{Q}_{r_1}, \mathcal{Q}_{r_2}, \ldots$. That is, we can use the logical qubit of the constant-size concatenated 7-qubit code in place of each physical qubit of $\mathcal{Q}^{(L)}$. With this modification, even if the physical error rate p_0 is larger than the threshold $p_{\rm th}$ for $\mathcal{Q}^{(L)}$ itself, the concatenated 7-qubit code can reduce the logical error rate from p_0 to a rate below $p_{\rm th}$, so that the rest of the concatenation $\mathcal{Q}_{r_1}, \mathcal{Q}_{r_2}, \ldots$ can arbitrarily suppress the logical error rate within the constant space overhead. In this way, the threshold of concatenating $\mathcal{Q}_3, \ldots, \mathcal{Q}_3, \mathcal{Q}_{r_1}, \mathcal{Q}_{r_2}, \ldots$ reduces to that of the concatenated 7-qubit code, with only increasing a constant factor of the space overhead for concatenating $\mathcal{Q}_3$ constant times.

Secondly, we can use $\mathcal{Q}_3$ in this concatenation as the error-detecting code to detect two errors, rather than the error-correcting code to correct one error. To use $\mathcal{Q}_3$ as the error-detecting code, we discard the states whenever we detect any error in the decoder. As long as this post-selection for error detection is performed only during the first constant times of the concatenation, the overhead of the post-selections is still constant. The post-selection techniques combined with the concatenated code can conventionally achieve *state-of-the-art high thresholds* [8, 34, 35], which can be even better than other leading candidates such as the topological codes.

Finally, we can also concatenate an *arbitrary* quantum code and $\mathcal{Q}^{(L)}$ as long as logical operations that we use in place of physical operations for our fault-tolerant protocol at concatenation level 0 are implementable for such a quantum code. For example, the surface code has well-established procedures for implementing logical operations for universal quantum computation [55, 56];

thus, we can use the logical qubit of a constant-size surface code in place of each physical qubit of $\mathcal{Q}^{(L)}$ to achieve the same threshold as that of the surface code, and at the same time attain the constant overhead asymptotically. Indeed, any quantum code with one logical qubit can be concatenated with $\mathcal{Q}^{(L)}$ by using the logical qubit of the code in place of each physical qubit of $\mathcal{Q}^{(L)}$, as long as the code can implement a set of logical operations that our fault-tolerant protocol uses on physical qubits, namely, a measurement in the Z basis, the H, S, CNOT, CZ, Pauli, and $R_y(\pm\pi/4)$ gates, and preparation of $|0\rangle$.

Given the fact that current quantum technology is insufficient for realizing physical error rate below a threshold in a scalable way, the significance of these modifications of our protocol is to provide flexibility in the protocol design from the theoretical side to reduce the technological demands in terms of the threshold while maintaining the constant overhead. However, these modifications may increase the space and time overheads of the fault-tolerant protocol up to a constant factor. Therefore, depending on the physical error rate achievable in experiments, the protocol with these modifications should be optimized to balance the trade-off between the overhead and the threshold. The optimization of the protocol may require further assumptions on technological advances, which we leave for future work. The techniques developed here constitute a fundamental step for further optimization of the fault-tolerant protocols.

Appendix F: Overhead

In this section, we prove that our fault-tolerant protocol achieves the constant space overhead. Furthermore, we also clarify the time overhead of our protocol. In particular, recalling that the task of FTQC formulated in Sec. A is to simulate the given $W(M)$-qubit $D(M)$-depth original circuit within an error ϵ, we will show that the time overhead of our protocol is quasi-polylogarithmic, i.e.,

$$\exp\left(O\left(\log^2\left(\log\left(\frac{M}{\epsilon}\right)\right)\right)\right). \tag{F1}$$

Subsequently, we will also discuss how we can modify our protocol to improve the scaling of the time overhead to

$$\exp\left(O\left(\log\log\left(\frac{M}{\epsilon}\right)\log\log\log\left(\frac{M}{\epsilon}\right)\right)\right). \tag{F2}$$

Starting from the $W(M)$-qubit $D(M)$-depth original circuit, our protocol in Fig. C.1 uses $\lceil W(M)/K^{(L)}\rceil$ level-L registers to represent the $W(M)$ qubits and further adds $8+2=10$ auxiliary level-L registers per register; in total, the level-L circuit has

$$11\lceil W(M)/K^{(L)}\rceil \text{ level-}L \text{ registers.} \tag{F3}$$

The depth of the level-L circuit is bounded as follows. The $W(M)$-qubit $D(M)$-depth original circuit consists of $D(M)$ one-depth parts. For each of the one-depth parts, we perform the Clifford gates in this part using the level-L Clifford-gate abbreviations (C15) and perform $R_y(\pm\pi/4)$ gates in this part by the level-L $R_y(\pm\pi/4)$-gate abbreviations (C16). A single use of level-L Clifford-gate abbreviation can apply two-qubit Clifford gates only between qubits in a pair of level-L registers on which the abbreviation acts. To perform two-qubit Clifford gates to qubits in different pairs of the level-L registers, we use different level-L Clifford-gate abbreviations. To bound the required depth of the level-L circuit for applying all the combinations of Clifford gates in a one-depth part of the original circuit, we consider, for each one-depth part, a graph with $\lceil W(M)/K^{(L)}\rceil$ vertices, where each vertex represents one of the level-L registers, and an edge connects a pair of vertices if and only if a two-qubit Clifford gate in this one-depth part connects the qubits in the corresponding pair of level-L registers. Since each level-L register has $K^{(L)}$ qubits, the qubits in a level-L register may be connected with those in at most $K^{(L)}$ other level-L registers by two-qubit Clifford gates in each one-depth part. Thus, this graph has at most degree $K^{(L)}$; that is, the graph is $(K^{(L)}+1)$-edge colorable [57]. For all the pairs of the vertices connected by the edges in the same color, we can in parallel perform the level-L Clifford-gate abbreviations on the corresponding pairs of level-L registers. Thus, we can perform any possible combination of Clifford gates in each one-depth part of the original circuit by at most $(K^{(L)}+1)$ parallel uses of the level-L two-register Clifford-gate abbreviations in the level-L circuit. Furthermore, we can perform the $R_y(\pm\pi/4)$ gates in each one-depth part by one parallel use of the level-L $R_y(\pm\pi/4)$-gate abbreviations. As a result, from the $W(M)$-qubit $D(M)$-depth original circuit, we obtain the compiled level-L circuit on $11\lceil W(M)/K^{(L)}\rceil$ level-L registers composed of $O(K^{(L)}D(M))$ parallel uses of level-L abbreviations. Since each level-L abbreviation has at most $O(\log(N^{(L)}))$ depth due to (D36) and (D37), the depth of the level-L circuit is bounded by

$$O(K^{(L)}\log(N^{(L)})\times D(M)). \tag{F4}$$

The number of locations in the level-L circuit is

$$\begin{aligned} &11\lceil W(M)/K^{(L)}\rceil \times O(K^{(L)}\log(N^{(L)})\times D(M)) \\ &= O(W(M)D(M)\log(N^{(L)})) \\ &\leqq CW(M)D(M)L^2, \quad \text{for large } M, \end{aligned} \tag{F5}$$

where the last line follows from $N^{(L)} = \exp(O(L^2))$ in (B27), and C is a constant factor.

For the fixed target error ϵ in simulating the original circuit, we then derive the required concatenation level L_M. On one hand, to achieve the target error ϵ with (F5), the required logical error rate is

$$p_L \leqq \frac{\epsilon}{CW(M)D(M)L^2}. \tag{F6}$$

On the other hand, the result (E18) of our threshold analysis shows that, to achieve (F6), it suffices to choose

L such that

$$2^L \geqq \log_{p_{\rm th}/p_0}\left(\frac{CW(M)D(M)}{\epsilon/p_{\rm th}}\right) + 2\log_{p_{\rm th}/p_0}(L), \quad \text{(F7)}$$

i.e.,

$$L = \Omega\left(\log\left(\log\left(\frac{W(M)D(M)}{\epsilon}\right)\right)\right) \quad \text{(F8)}$$

$$= \Omega\left(\log\left(\log\left(\frac{M}{\epsilon}\right)\right)\right), \quad \text{(F9)}$$

where the last line follows from $\log(W(M)D(M)) = \log(\mathrm{poly}(M)) \approx \log(M)$. To keep the code $\mathcal{Q}^{(L)}$ used for our fault-tolerant protocol as small as possible, we here choose, up to a constant factor,

$$L_M = \Theta\left(\log\left(\log\left(\frac{M}{\epsilon}\right)\right)\right). \quad \text{(F10)}$$

For later convenience, the number of physical and logical qubits of $\mathcal{Q}_{r_{L_M}}$ grows, due to $N_{r_L} = \exp(O(L))$ in (B19) and $K_{r_L} = \exp(O(L))$ in (B20), as

$$N_{r_{L_M}} = \exp\left(O\left(\log\left(\log\left(\frac{M}{\epsilon}\right)\right)\right)\right) \quad \text{(F11)}$$

$$= O\left(\mathrm{polylog}\left(\frac{M}{\epsilon}\right)\right), \quad \text{(F12)}$$

$$K_{r_{L_M}} = \exp\left(O\left(\log\left(\log\left(\frac{M}{\epsilon}\right)\right)\right)\right) \quad \text{(F13)}$$

$$= O\left(\mathrm{polylog}\left(\frac{M}{\epsilon}\right)\right). \quad \text{(F14)}$$

On the other hand, due to $N^{(L)} = \exp(O(L^2))$ in (B27) and $K^{(L)} = \exp(O(L^2))$ in (B28), each level-L_M register has a quasi-polylogarithmic size in M, i.e.,

$$N^{(L_M)} = \exp\left(O\left(\log^2\left(\log\left(\frac{M}{\epsilon}\right)\right)\right)\right), \quad \text{(F15)}$$

$$K^{(L_M)} = \exp\left(O\left(\log^2\left(\log\left(\frac{M}{\epsilon}\right)\right)\right)\right). \quad \text{(F16)}$$

With the above choice of L_M, our protocol achieves the constant space overhead as shown in the following. Starting from the $11^{\lceil W(M)/K^{(L_M)}\rceil}$ level-L_M registers in (F3), we recursively replace each level-l register with $(N_{r_l} + 8 + 2) = (N_{r_l} + 10)$ level-$(l-1)$ registers as in (D33), except that we replace a level-1 register with N_{r_1} physical qubits without auxiliary ones. Thus, the total required number of physical qubits is given by

$$W_{\rm FT}(M) = 11^{\lceil W(M)/K^{(L_M)}\rceil} \times \left(N_{r_1}\prod_{l=2}^{L_M}(N_{r_l}+10)\right); \quad \text{(F17)}$$

i.e., we have

$$\frac{W_{\rm FT}(M)}{W(M)} \quad \text{(F18)}$$

$$= \frac{11^{\lceil W(M)/K^{(L_M)}\rceil} \times (N_{r_1}\prod_{l=2}^{L_M}(N_{r_l}+10))}{W(M)} \quad \text{(F19)}$$

$$\leqq \frac{11^{(W(M)/K^{(L_M)}+1)} \times (N_{r_1}\prod_{l=2}^{L_M}(N_{r_l}+10))}{W(M)} \quad \text{(F20)}$$

$$= 11\left(\frac{N_{r_1}}{K_{r_1}}\prod_{l=2}^{L_M}\frac{N_{r_l}+10}{K_{r_l}}\right)\left(1+\frac{K^{(L_M)}}{W(M)}\right) \quad \text{(F21)}$$

$$\leqq 11\left(\frac{N_{r_1}}{K_{r_1}}\prod_{l=2}^{L_M}\frac{N_{r_l}+10}{K_{r_l}}\right)\left(1+\frac{K^{(L_M)}}{M}\right), \quad \text{(F22)}$$

where the last line follows from $W(M) \geqq M$ in (A2). The factor $\left(1 + K^{(L_M)}/M\right)$ in (F22) converges to 1 as $M \to \infty$ since $K^{(L_M)}$ is quasi-polylogarithmic in M as shown in (F16). To see that the dominant factor in (F22), i.e.,

$$11\left(\frac{N_{r_1}}{K_{r_1}}\prod_{l=2}^{L_M}\frac{N_{r_l}+10}{K_{r_l}}\right) \quad \text{(F23)}$$

is $O(1)$ as $M \to \infty$, recall that the crucial condition in deriving the non-vanishing rate (B35) of $\mathcal{Q}^{(L)}$ has been the convergence of the infinite sum (B34). Since this condition also holds in the case here, i.e.,

$$\sum_{l=2}^{\infty}\left|\frac{(N_{r_l}+10)-K_{r_l}}{K_{r_l}}\right| < \infty, \quad \text{(F24)}$$

in the same way as showing (B35), it holds that

$$\prod_{l=2}^{L_M}\frac{N_{r_l}+10}{K_{r_l}} = O(1), \quad \text{as } M \to \infty. \quad \text{(F25)}$$

Therefore, the space overhead (A6) is

$$\frac{W_{\rm FT}(M)}{W(M)} = O(1), \quad \text{as } M \to \infty, \quad \text{(F26)}$$

achieving the constant space overhead.

For concreteness, in Fig. F.1, we numerically evaluate the dominant factor (F23) of the space overhead in (F22). The figure shows that even if we include all the auxiliary qubits, the space overhead of our fault-tolerant protocol at each time step is upper bounded by the factor of 1163 as a whole on arbitrarily large scales. Note that the significance of our work is to develop the fundamental techniques for our protocol to achieve the constant space overhead, and we leave further optimization of the constant factor for future research.

Regarding the number of bits, our protocol is designed to use $O(\mathrm{poly}(N^{(l)}))$ bits in a level-l gadget for any l, as in (D34). The level-l circuits for $l \in \{L_M, L_M-1, \ldots, 0\}$ in our protocol are composed of $O(\mathrm{poly}(M))$ elementary operations. Therefore, the number of bits to store all data used in the gadgets in these circuits at all the concatenation levels is upper bounded by

$$\sum_{l=1}^{L_M} O(\mathrm{poly}(N^{(l)})) \times O(\mathrm{poly}(M)) = O(\mathrm{poly}(M)), \quad \text{(F27)}$$

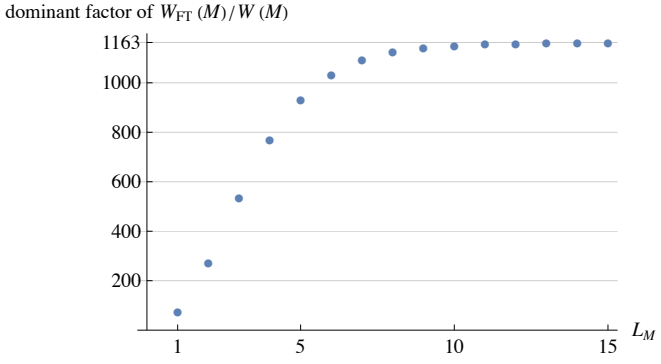


FIG. F.1. The dominant factor of the space overhead (i.e., (F23)) in our fault-tolerant protocol including all the auxiliary qubits used at a time. The space overhead at each time step is upper bounded by the factor of 1163 for an arbitrarily large concatenation level L_M while the doubly exponential suppression of the logical error rate is achieved as L_M increases.

satisfying (A7).

As for the number of parallel processes of classical computation, a level-l gadget uses at most $O(\mathrm{poly}(N^{(l)}))$ processes at a time, as in (D25). Moreover, the level-l gadget is implemented using at most $O(N_{r_l})$ level-$(l-1)$ elementary operations at a time, where each level-$(l-1)$ gadget for a level-$(l-1)$ elementary operation uses at most $O(\mathrm{poly}(N^{(l-1)}))$ processes at a time. By repeating this counting recursively, the number of processes for implementing each level-L_M operation is

$$\begin{aligned}
&O(\mathrm{poly}(N^{(L_M)}))+\\
&\quad O(N_{r_{L_M}})\times O(\mathrm{poly}(N^{(L_M-1)}))+\\
&\quad O(N_{r_{L_M}}N_{r_{L_M-1}})\times O(\mathrm{poly}(N^{(L_M-2)}))+\cdots\\
&\quad O(N_{r_{L_M}}N_{r_{L_M-1}}\cdots N_{r_2})\times O(\mathrm{poly}(N^{(1)})) \qquad \text{(F28)}\\
&=O(\mathrm{poly}(N^{(L_M)})). \qquad \text{(F29)}
\end{aligned}$$

Since we have $O(W(M)/K^{(L_M)})$ level-L_M registers at a time, the number of parallel processes used at a time is bounded by

$$\begin{aligned}
&O\left(\frac{W(M)}{K^{(L_M)}}\times\mathrm{poly}(N^{(L_M)})\right) \qquad \text{(F30)}\\
&=W(M)\times\exp(O(L_M^2)) \qquad \text{(F31)}\\
&=W(M)\times\exp\left(O\left(\log^2\left(\log\left(\frac{M}{\epsilon}\right)\right)\right)\right), \qquad \text{(F32)}
\end{aligned}$$

satisfying (A8).

As for the time overhead, for each $l=L_M,\dots,1$, each level-l operation is replaced with an $O(\mathrm{poly}(N_{r_l}))$-depth level-$(l-1)$ circuit, as in (D23) (see also the counting in deriving (E5)). Thus, in terms of the depth, the level-$(l-1)$ circuit is at most $O(\mathrm{poly}(N_{r_l}))$ times as deep as the level-l circuit. Therefore, starting from the $O(K^{(L_M)}\log(N^{(L_M)})D(M))$-depth level-$L_M$ circuit in (F4), we multiply $O(\mathrm{poly}(N_{r_l}))$ recursively to obtain the depth of the resulting level-0 circuit as

$$\begin{aligned}
&D_{\rm FT}(M)\\
&=O\left(K^{(L_M)}\log(N^{(L_M)})D(M)\times\prod_{l=1}^{L_M}\mathrm{poly}(N_{r_l})\right) \qquad \text{(F33)}\\
&=D(M)\times\left(\exp(O(L_M^2))\times\prod_{l=1}^{L_M}\exp(O(l))\right) \qquad \text{(F34)}\\
&=D(M)\times\exp(O(L_M^2)) \qquad \text{(F35)}\\
&=D(M)\times\exp\left(O\left(\log^2\left(\log\left(\frac{M}{\epsilon}\right)\right)\right)\right). \qquad \text{(F36)}
\end{aligned}$$

As a result, the time overhead (A9) is

$$\frac{D_{\rm FT}(M)}{D(M)}=\exp\left(O\left(\log^2\left(\log\left(\frac{M}{\epsilon}\right)\right)\right)\right), \qquad \text{(F37)}$$

achieving the quasi-polylogarithmic time overhead in M. The conclusion in (F10), (F26), and (F37) is summarized as Proposition 2 in the main text.

We remark that the above argument on the space and time overhead works fairly in general. In particular, the same argument leads to the following theorem.

Theorem 3 (Space and time overhead)**.** Under the assumptions of Theorems 1 and 2, if the fault-tolerant protocol of Theorem 2 uses at most $n_l+O(1)$ level-$(l-1)$ registers per encoded level-l register, then for any $W(M)=O(\mathrm{poly}(M))$ and $D(M)=O(\mathrm{poly}(M))$, there exists a concatenation level

$$L_M=O\left(\log\left(\log\left(\frac{M}{\epsilon}\right)\right)\right) \qquad \text{(F38)}$$

such that the fault-tolerant protocol can simulate any $W(M)$-qubit $D(M)$-depth original circuit within error $\epsilon>0$ in total variational distance with constant space overhead

$$\frac{W_{\rm FT}(M)}{W(M)}=O(1) \qquad \text{(F39)}$$

and quasi-polylogarithmic time overhead

$$\frac{D_{\rm FT}(M)}{D(M)}=\exp\left(O\left(\mathrm{polylog}\left(\log\left(\frac{M}{\epsilon}\right)\right)\right)\right). \qquad \text{(F40)}$$

In particular, instead of taking the parameter r_l of the code $\mathcal{Q}_{r_l}$ linearly in l, i.e., $r_l=l+2$ as we have done in (B18), we could grow r_l more slowly to make the scaling of the time overhead smaller. For example, for any constant $\beta>1$, choose

$$r_l=\lceil\beta\log_2(l)\rceil. \qquad \text{(F41)}$$

With the choice (F41), the number of physical qubits of $\mathcal{Q}_{r_l}$ in (B4) and that of logical qubits in (B5) are given respectively by

$$N_{r_l}=2^{\lceil\beta\log_2(l)\rceil}-1, \qquad \text{(F42)}$$

$$K_{r_l} = 2^{\lceil \beta \log_2(l) \rceil} - 2\lceil \beta \log_2(l) \rceil - 1. \quad \text{(F43)}$$

Thus, even if we replace $r_l = l + 2$ with $r_l = \lceil \beta \log_2(l) \rceil$, we can still see that the infinite sum (F24) converges as

$$\sum_{l=2}^{\infty} \left| \frac{(N_{r_l} + 10) - K_{r_l}}{N_{r_l}} \right| = \sum_{l=2}^{\infty} O\left(\frac{\log(l)}{l^\beta} \right) < \infty, \quad \text{(F44)}$$

which leads to (F25). Therefore, the choice (F41) still achieves the constant-space-overhead FTQC, i.e.,

$$\frac{W_{\mathrm{FT}}(M)}{W(M)} = O(1). \quad \text{(F45)}$$

As for the time overhead, due to (B25) and (B26), the number of physical qubits of $\mathcal{Q}^{(l)}$ and that of logical qubits scale respectively as

$$N^{(l)} = \prod_{l'=1}^{l} N_{r_{l'}} = \exp(O(l \log(l))), \quad \text{(F46)}$$

$$K^{(l)} = \prod_{l'=1}^{l} K_{r_{l'}} = \exp(O(l \log(l))). \quad \text{(F47)}$$

Thus, the level-l gadgets with $r_l = \lceil \beta \log_2(l) \rceil$ are smaller than those with $r_l = l + 2$ in terms of the scaling of the sizes; therefore, the scaling (E18) of the logical error rate derived in our analysis is still valid, and we can choose the overall concatenation level L_M in the same way as (F10). Applying these scalings to (F33), we achieve the time overhead scaling as

$$\frac{D_{\mathrm{FT}}(M)}{D(M)} = \exp\left(O\left(\log\log\left(\frac{M}{\epsilon} \right) \log\log\log\left(\frac{M}{\epsilon} \right) \right) \right). \quad \text{(F48)}$$

As a result, our protocol with the modification of (F41) can achieve a time overhead that is polylogarithmic up to the $\log\log\log(M/\epsilon)$ factor in the exponent. Note that, while improving the asymptotic scaling of the time overhead, this modification of our protocol may increase the constant factor of the space overhead. The overall optimization of the protocol to balance the constant factor and the asymptotic scaling is left for future work.

For comparison, the existing constant-space-overhead fault-tolerant protocol [3, 6, 7] incurs a polynomially large time overhead even though the existing protocol assumes that classical computation during the protocol can be performed instantaneously within zero time. In particular, the existing protocol uses a concatenated code to prepare encoded auxiliary states used for gate teleportation, but due to the polylogarithmic space overhead in using the conventional concatenated code, the encoded auxiliary states cannot be prepared for applying the gates on all the code blocks in parallel at a time [3]. Since the size of each code block in the existing protocol is bounded polynomially by $O(M^\alpha)$ for fixed $\alpha > 0$, the gates are applicable only to a polynomially small fraction of the code blocks within the constant space overhead; hence, the existing protocol incurs the polynomially large time overhead unless one improves the analysis in Ref. [3] to reduce the required size of code blocks for a desired error suppression. In addition, for the fair comparison with our protocol, the time overhead of the constant-space-overhead protocol using quantum LDPC codes also needs to be analyzed further by taking into account the wait operations to wait for the nonzero runtime of classical computation since our protocol is formulated and analyzed in this more demanding but practical setting. The existing analysis of the constant-space-overhead protocol using quantum LDPC codes [3, 6, 7] does not take into account such nonzero runtime of classical computation. As discussed in Sec. E, to prove the existence of a threshold in the same setting as our protocol, one also needs to develop further new techniques beyond Ref. [3] to deal with nonzero runtime of classical computation in the decoder of the quantum LDPC code. Despite this more demanding setting, our analysis proves that our fault-tolerant protocol achieves the constant space overhead and the quasi-polylogarithmic time overhead at the same time. Owing to the doubly exponential error suppression as the concatenation level increases, we can keep the required code size small, and hence, our protocol significantly improves the time overhead, from polynomial to quasi-polylogarithmic. It would be an interesting open question whether a polylogarithmic time overhead or even a sub-polylogarithmic time overhead is achievable within the constant space overhead under reasonable assumptions including the nonzero runtime of classical computation in conducting FTQC.

[1] Gottesman, D. An introduction to quantum error correction and fault-tolerant quantum computation. In *Quantum information science and its contributions to mathematics*, vol. 68 of *Proceedings of Symposia in Applied Mathematics*, 13–58 (American Mathematical Society, Providence, Rhode Island, 2010). URL http://www.ams.org/books/psapm/068/2762145.

[2] Nielsen, M. A. & Chuang, I. L. *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, Cambridge, 2010). URL https://www.cambridge.org/highereducation/books/quantum-computation-and-quantum-information/01E10196D0A682A6AEFFEA52D53BE9AE.

[3] Gottesman, D. Fault-tolerant quantum computation

with constant overhead. *Quantum Info. Comput.* **14**, 1338–1372 (2014). URL https://www.rintonpress.com/journals/doi/QIC14.15-16-5.html.

[4] Reischuk, R. & Schmeltz, B. Reliable computation with noisy circuits and decision trees-a general n log n lower bound. In *[1991] Proceedings 32nd Annual Symposium of Foundations of Computer Science*, 602–611 (1991). URL https://ieeexplore.ieee.org/document/185425.

[5] Romashchenko, A. Reliable computations based on locally decodable codes. In Durand, B. & Thomas, W. (eds.) *STACS 2006*, 537–548 (Springer Berlin Heidelberg, Heidelberg, 2006). URL https://link.springer.com/chapter/10.1007/11672142_44.

[6] Kovalev, A. A. & Pryadko, L. P. Fault tolerance of quantum low-density parity check codes with sublinear distance scaling. *Phys. Rev. A* **87**, 020304 (2013). URL https://link.aps.org/doi/10.1103/PhysRevA.87.020304.

[7] Fawzi, O., Grospellier, A. & Leverrier, A. Constant overhead quantum fault tolerance with quantum expander codes. *Commun. ACM* **64**, 106–114 (2020). URL https://doi.org/10.1145/3434163.

[8] Yamasaki, H., Fukui, K., Takeuchi, Y., Tani, S. & Koashi, M. Polylog-overhead highly fault-tolerant measurement-based quantum computation: all-gaussian implementation with gottesman-kitaev-preskill code (2020). URL https://arxiv.org/abs/2006.05416.

[9] Yamasaki, H., Matsuura, T. & Koashi, M. Cost-reduced all-gaussian universality with the gottesman-kitaev-preskill code: Resource-theoretic approach to cost analysis. *Phys. Rev. Research* **2**, 023270 (2020). URL https://link.aps.org/doi/10.1103/PhysRevResearch.2.023270.

[10] Baragiola, B. Q., Pantaleoni, G., Alexander, R. N., Karanjai, A. & Menicucci, N. C. All-gaussian universality and fault tolerance with the gottesman-kitaev-preskill code. *Phys. Rev. Lett.* **123**, 200502 (2019). URL https://link.aps.org/doi/10.1103/PhysRevLett.123.200502.

[11] Menicucci, N. C. Fault-tolerant measurement-based quantum computing with continuous-variable cluster states. *Phys. Rev. Lett.* **112**, 120504 (2014). URL https://link.aps.org/doi/10.1103/PhysRevLett.112.120504.

[12] Takeda, S. & Furusawa, A. Toward large-scale fault-tolerant universal photonic quantum computing. *APL Photonics* **4**, 060902 (2019). URL https://aip.scitation.org/doi/10.1063/1.5100160.

[13] Yamasaki, H. *Entanglement theory in distributed quantum information processing*. Ph.D. thesis, The University of Tokyo (2019). URL https://arxiv.org/abs/1903.09655.

[14] Wehner, S., Elkouss, D. & Hanson, R. Quantum internet: A vision for the road ahead. *Science* **362**, eaam9288 (2018). URL https://www.science.org/doi/abs/10.1126/science.aam9288.

[15] Meter, R. V. & Devitt, S. J. The path to scalable distributed quantum computing. *Computer* **49**, 31–42 (2016). URL https://ieeexplore.ieee.org/document/7562346.

[16] Aharonov, D. & Ben-Or, M. Fault-tolerant quantum computation with constant error rate. *SIAM J. Comput.* **38**, 1207–1282 (2008). URL https://doi.org/10.1137/S0097539799359385.

[17] Gottesman, D. Fault-tolerant quantum computation with local gates. *Journal of Modern Optics* **47**, 333–345 (2000). URL https://www.tandfonline.com/doi/abs/10.1080/09500340008244046.

[18] Svore, K. M., Terhal, B. M. & DiVincenzo, D. P. Local fault-tolerant quantum computation. *Phys. Rev. A* **72**, 022317 (2005). URL https://link.aps.org/doi/10.1103/PhysRevA.72.022317.

[19] Svore, K. M., DiVincenzo, D. P. & Terhal, B. M. Noise threshold for a fault-tolerant two-dimensional lattice architecture. *Quantum Inf. Comput.* **7**, 297–318 (2007). URL https://doi.org/10.26421/QIC7.4-2.

[20] Preskill, J. Quantum error correction (2020). URL http://theory.caltech.edu/~preskill/ph229/. Chapter 7 of Lecture Notes for Physics 219/Computer Science 219 Quantum Computation (Formerly Physics 229).

[21] Devitt, S. J., Munro, W. J. & Nemoto, K. Quantum error correction for beginners. *Rep. Prog. Phys.* **76**, 076001 (2013). URL http://iopscience.iop.org/article/10.1088/0034-4885/76/7/076001.

[22] Terhal, B. M. Quantum error correction for quantum memories. *Rev. Mod. Phys.* **87**, 307–346 (2015). URL https://link.aps.org/doi/10.1103/RevModPhys.87.307.

[23] Lidar, D. A. & Brun, T. A. *Quantum Error Correction* (Cambridge University Press, Cambridge, 2013). URL https://www.cambridge.org/core/books/quantum-error-correction/B51E8333050A0F9A67363254DC1EA15A.

[24] Lint, J. H. V. *Introduction to Coding Theory* (Springer-Verlag Berlin, Heidelberg, 1998), 3rd edn. URL https://link.springer.com/book/10.1007/978-3-642-58575-3.

[25] Hamming, R. W. Error detecting and error correcting codes. *The Bell System Technical Journal* **29**, 147–160 (1950). URL https://ieeexplore.ieee.org/document/6772729.

[26] Steane, A. M. Simple quantum error-correcting codes. *Phys. Rev. A* **54**, 4741–4751 (1996). URL https://link.aps.org/doi/10.1103/PhysRevA.54.4741.

[27] Steane, A. M. Error correcting codes in quantum theory. *Phys. Rev. Lett.* **77**, 793–797 (1996). URL https://link.aps.org/doi/10.1103/PhysRevLett.77.793.

[28] Steane, A. Multiple-particle interference and quantum error correction. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* **452**, 2551–2577 (1996). URL https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1996.0136.

[29] Gottesman, D. *Stabilizer Codes and Quantum Error Correction*. Ph.D. thesis, California Institute of Technology (1997). URL https://arxiv.org/abs/quant-ph/9705052.

[30] Wilde, M. M. Logical operators of quantum codes. *Phys. Rev. A* **79**, 062322 (2009). URL https://link.aps.org/doi/10.1103/PhysRevA.79.062322.

[31] Little, C. H. C., Teo, K. L. & van Brunt, B. *An Introduction to Infinite Products* (Springer International Publishing, Cham, 2022). URL https://link.springer.com/book/10.1007/978-3-030-90646-7.

[32] Aaronson, S. & Gottesman, D. Improved simulation of stabilizer circuits. *Phys. Rev. A* **70**, 052328 (2004). URL https://link.aps.org/doi/10.1103/PhysRevA.70.052328.

[33] Gottesman, D. & Chuang, I. L. Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. *Nature* **402**, 390 – 393 (1999). URL https://www.nature.com/articles/46503.

[34] Knill, E. Quantum computing with realistically noisy devices. *Nature* **434**, 39 – 44 (2005). URL https://www.nature.com/articles/nature03350.

[35] Knill, E. Scalable quantum computing in the presence of large detected-error rates. *Phys. Rev. A* **71**, 042322 (2005). URL https://link.aps.org/doi/10.1103/PhysRevA.71.042322.

[36] Bennett, C. H. *et al.* Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels. *Phys. Rev. Lett.* **70**, 1895–1899 (1993). URL https://link.aps.org/doi/10.1103/PhysRevLett.70.1895.

[37] Cleve, R. & Gottesman, D. Efficient computations of encodings for quantum error correction. *Phys. Rev. A* **56**, 76–82 (1997). URL https://link.aps.org/doi/10.1103/PhysRevA.56.76.

[38] Paetznick, A. & Reichardt, B. W. Fault-tolerant ancilla preparation and noise threshold lower boudds for the 23-qubit golay code. *Quantum Info. Comput.* **12**, 1034–1080 (2012). URL https://www.rintonpress.com/journals/doi/QIC12.11-12-10.html.

[39] Steane, A. M. Fast fault-tolerant filtering of quantum codewords (2002). URL https://arxiv.org/abs/quant-ph/0202036.

[40] Moore, C. & Nilsson, M. Parallel quantum computation and quantum codes. *SIAM J. Comput.* **31**, 799–815 (2002). URL https://doi.org/10.1137/S0097539799355053.

[41] Jiang, J. *et al.* *Optimal Space-Depth Trade-Off of CNOT Circuits in Quantum Logic Synthesis*, 213–229 (2020). URL https://epubs.siam.org/doi/abs/10.1137/1.9781611975994.13.

[42] Chao, R. & Reichardt, B. W. Quantum error correction with only two extra qubits. *Phys. Rev. Lett.* **121**, 050502 (2018). URL https://link.aps.org/doi/10.1103/PhysRevLett.121.050502.

[43] Maslov, D. & Roetteler, M. Shorter stabilizer circuits via bruhat decomposition and quantum circuit transformations. *IEEE Transactions on Information Theory* **64**, 4729–4738 (2018). URL https://ieeexplore.ieee.org/document/8335339.

[44] Patel, K. N., Markov, I. L. & Hayes, J. P. Optimal synthesis of linear reversible circuits. *Quantum Info. Comput.* **8**, 282–294 (2008). URL https://www.rintonpress.com/journals/doi/QIC8.3-4-4.html.

[45] Goto, H. Minimizing resource overheads for fault-tolerant preparation of encoded states of the steane code. *Sci. Rep.* **6**, 19578 (2016). URL https://www.nature.com/articles/srep19578.

[46] Chamberland, C. & Cross, A. W. Fault-tolerant magic state preparation with flag qubits. *Quantum* **3**, 143 (2019). URL https://doi.org/10.22331/q-2019-05-20-143.

[47] Holmes, A. *et al.* Nisq+: Boosting quantum computing power by approximating quantum error correction. In *Proceedings of the ACM/IEEE 47th Annual International Symposium on Computer Architecture*, ISCA '20, 556–569 (IEEE Press, 2020). URL https://doi.org/10.1109/ISCA45697.2020.00053.

[48] Chamberland, C., Goncalves, L., Sivarajah, P., Peterson, E. & Grimberg, S. Techniques for combining fast local decoders with global decoders under circuit-level noise (2022). URL https://arxiv.org/abs/2208.01178.

[49] Skoric, L., Browne, D. E., Barnes, K. M., Gillespie, N. I. & Campbell, E. T. Parallel window decoding enables scalable fault tolerant quantum computation (2022). URL https://arxiv.org/abs/2209.08552.

[50] Tan, X., Zhang, F., Chao, R., Shi, Y. & Chen, J. Scalable surface code decoders with parallelization in time (2022). URL https://arxiv.org/abs/2209.09219.

[51] Bombín, H. *et al.* Modular decoding: parallelizable real-time decoding for quantum computers (2023). URL https://arxiv.org/abs/2303.04846.

[52] Grospellier, A. *Constant time decoding of quantum expander codes and application to fault-tolerant quantum computation.* Ph.D. thesis, Sorbonne Universié (2019). URL https://tel.archives-ouvertes.fr/tel-03364419/document.

[53] Ueno, Y., Kondo, M., Tanaka, M., Suzuki, Y. & Tabuchi, Y. Qecool: On-line quantum error correction with a superconducting decoder for surface code. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 451–456 (2021). URL https://ieeexplore.ieee.org/document/9586326.

[54] Bourassa, J. E. *et al.* Blueprint for a Scalable Photonic Fault-Tolerant Quantum Computer. *Quantum* **5**, 392 (2021). URL https://doi.org/10.22331/q-2021-02-04-392.

[55] Horsman, C., Fowler, A. G., Devitt, S. & Meter, R. V. Surface code quantum computing by lattice surgery. *New Journal of Physics* **14**, 123011 (2012). URL https://doi.org/10.1088/1367-2630/14/12/123011.

[56] Litinski, D. A Game of Surface Codes: Large-Scale Quantum Computing with Lattice Surgery. *Quantum* **3**, 128 (2019). URL https://doi.org/10.22331/q-2019-03-05-128.

[57] Bondy, J. A. & Murty, U. S. R. *Graph Theory*, vol. 244 of *Graduate Texts in Mathematics* (Springer London, London, 2008). URL https://link.springer.com/gp/book/9781846289699.