Correlated decoding of logical algorithms with transversal gates

Madelyn Cain[1], Chen Zhao[1,2], Hengyun Zhou[1,2], Nadine Meister[1], J. Pablo Bonilla Ataides[1], Arthur Jaffe[1], Dolev Bluvstein[1], and Mikhail D. Lukin[1]
[1] *Department of Physics, Harvard University, Cambridge, MA 02138, USA*
[2] *QuEra Computing Inc., Boston, MA 02135, USA*

(Dated: March 7, 2024)

Quantum error correction is believed to be essential for scalable quantum computation, but its implementation is challenging due to its considerable space-time overhead. Motivated by recent experiments demonstrating efficient manipulation of logical qubits using transversal gates (Bluvstein *et al.*, Nature **626**, 58-65 (2024)), we show that the performance of logical algorithms can be substantially improved by decoding the qubits jointly to account for physical error propagation during transversal entangling gates. We find that such correlated decoding improves the performance of both Clifford and non-Clifford transversal entangling gates, and explore two decoders offering different computational runtimes and accuracies. By considering deep logical Clifford circuits, we find that correlated decoding can significantly improve the space-time cost by reducing the number of rounds of noisy syndrome extraction per gate. These results demonstrate that correlated decoding provides a major advantage in early fault-tolerant computation, and indicate it has considerable potential to reduce the space-time cost in large-scale logical algorithms.

Quantum error correction (QEC) is believed to be essential for large-scale quantum computation [1–5]. It is a rapidly developing frontier, with recent milestone demonstrations of the preservation of a logical qubit and universal logic gates [6–16]. However, implementing large-scale algorithms with protected, logical qubits is challenging due to its considerable space-time overhead. Most recently, reconfigurable neutral atom arrays have realized quantum algorithms with tens of logical qubits and hundreds of logical entangling gates [17], opening the door to early fault-tolerant computation. Key to these results was the ability to efficiently perform fault-tolerant transversal entangling gates by dynamically reconfiguring qubits during the computation [18], similar to prior work with trapped ions [9, 10, 13, 15].

These experiments [17] motivate theoretical explorations of the optimal ways to design and decode error-corrected circuits, in order to accurately infer which physical errors occurred such that they can be corrected [19, 20]. In particular, transversal entangling gates apply the same operation to the physical qubits within the codes, resulting in highly structured error propagation: errors spread in a known way between, but not within, logical code blocks. As a result, errors detected on one logical qubit can contain information about which errors occurred on other logical qubits, and these correlations can be utilized to improve the accuracy of the decoder. Such correlated decoding was already used in Ref. [17], demonstrating its utility for near-term experiments. In addition, it may impact the resource requirements of complex large-scale quantum algorithms [21–23], as transversal gates can comprise core subroutines in both QEC [24, 25] and algorithms [26].

In this Letter, we analyze the benefits of jointly decoding the logical qubits in an algorithm to account for error propagation during transversal entangling gates. We explore two decoders which utilize the space-time decoding hypergraph of the logical algorithm [27–32], one of which is highly accurate, and another which is approximate but guaranteed to have an efficient runtime. We find that correlated decoding improves the performance of both Clifford and non-Clifford transversal entangling gates. We also show that it enables the number of syndrome extraction rounds to be substantially reduced in deep logical Clifford circuits. These results provide the theoretical foundation for recent experimental observations [17], highlight the utility of correlated decoding in early fault-tolerant computation, and indicate its considerable potential to reduce the space-time cost of large-scale logical algorithms.

The key idea of this work can be understood as follows. We consider a quantum algorithm with logical qubits, each comprised of redundant physical qubits [3, 33] (see Fig. 1(a)). The logical information of each code is stored in the simultaneous +1 eigenspace of its stabilizer operators. Physical errors can be detected by measuring the stabilizers, as any error which anticommutes with a stabilizer will flip its measurement outcome to −1. A decoding algorithm can use these stabilizer measurements, or syndromes, to predict which physical error(s) likely caused the observed syndromes. Decoding succeeds if the associated correction does not erroneously apply a logical operation.

Errors during transversal entangling gates can create correlations in the stabilizer measurements of different logical qubits, which can be utilized to improve the accuracy of the decoder. In a transversal CNOT, for example, physical X errors are copied from the control to the target qubit, and Z errors are copied from the target to the control qubit (Fig. 1(a)). Although this propagation increases the density of errors, it is deterministic and can be deduced from correlations in the measured

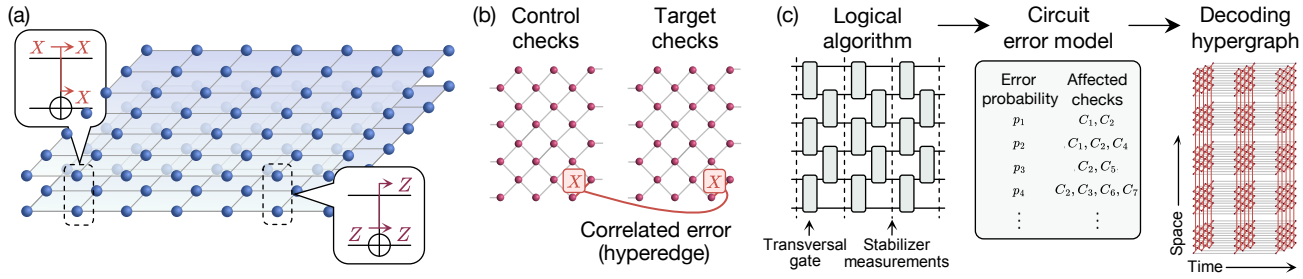

FIG. 1. Decoding logical algorithms with transversal gates. (a) Transversal CNOT gates copy physical errors between logical qubits. (b) These transferred errors flip the same stabilizers on both logical qubits after the CNOT, generating a hyperedge in the decoding hypergraph connecting the control and target check vertices. (c) Given a physical error model, the decoding hypergraph for a logical algorithm can track these transferred errors, which can appear as hyperedges connecting checks from multiple logical qubits at different points in time.

stabilizers. For instance, assuming that X errors occur only before a transversal CNOT, one can infer that a flipped Z stabilizer on both logical qubits originates from a single X error copied from the control, effectively halving the density of errors in the decoding. Similarly, a physical error can propagate onto N logical qubits after N sequential CNOTs. Although this process flips many stabilizers, tracking the error propagation again reveals that the syndrome originated from just a single error.

Correlated decoding algorithms.— To formalize this approach, we define the decoding hypergraph of the logical algorithm [27–31], which relates each possible physical error mechanism to the stabilizer measurement(s) it flips. The hypergraph vertices correspond to N measured *checks* of the logical qubits $\{C_i\}_{i=1\ldots N}$, which compare consecutive stabilizer measurements in time. Each check is defined as the product of a stabilizer measurement and its previous measurement (if one exists), obtained from back-propagating the stabilizer operator through the circuit to the point it was previously measured. If one stabilizer measurement in the check detects an error, the check flips from $+1$ to -1. Each error mechanism is associated with a hyperedge connecting the check(s) it flips. Errors copied during transversal entangling gates therefore correspond to hyperedges connecting the checks of multiple logical qubits (Fig. 1(b)). We use Stim [27], a Clifford circuit simulator, to identify the M hyperedges $\{E_j\}_{j=1\ldots M}$ and their probabilities $\{p_j\}_{j=1\ldots M}$ under a chosen circuit error model, and which error source(s) $I(C_i)$ flip each check C_i (Fig. 1(c)).

In order to realize QEC in practice, a specific decoder should be chosen. Many conventional decoders such as minimum weight perfect matching (MWPM) [34, 35], a widely used surface code decoder, are constrained to operate on decoding *graphs*, whose edges connect at most two vertices. In contrast, here we explore two decoders which take as input a general decoding hypergraph of a logical algorithm, which can include hyperedges connecting more than two vertices. The first decoder exactly computes the most-likely error (MLE) consistent with the measured syndrome. It maximizes the total error probability $\prod_{j=1}^{M} p_j^{E_j}(1-p_j)^{1-E_j}$ over the binary variables E_j, which are equal to one if the error E_j occurred and zero otherwise. The errors are constrained to be consistent with the measured syndrome: if $C_i = +1$ (-1), then an even (odd) number of errors occurred in $I(C_i)$. We map this problem to a mixed-integer program, and find its optimal solution using a state-of-the-art solver [32, 36–38].

Because finding the MLE exactly is NP-hard, we expect the runtime of the above algorithm to grow exponentially in M in the worst-case [39, 40]. This motivates our second algorithm, hypergraph union-find with belief propagation (belief-HUF) [41, 42], which instead approximates the MLE and has an efficient time complexity of $O(M^3 \log(M))$ [32]. Belief-HUF first uses belief propagation to update the error probabilities $\{p_j\}_{j=1\ldots M}$ to approximate the probability of each error given the measured syndrome [29]. Leveraging the insight that errors are typically localized, belief-HUF then expands clusters of error mechanisms and checks on the decoding hypergraph at a rate dependent on the error probabilities, merging clusters that meet. A cluster stops expanding when it contains an error mechanism consistent with its measured checks, and the algorithm terminates when all clusters stop expanding.

Decoding transversal Clifford gates.— As a simple illustrative example, we apply these techniques to decode a transversal CNOT, which is available for all Calderbank-Shor-Steane (CSS) codes [43]. A key property of error-correcting circuits is that their performance can be improved for physical error rates below some threshold p_{th} by increasing the code distance (the minimum weight of a logical Pauli operator), which is related to how many errors can be corrected [3, 25]. The threshold fundamentally depends on the decoder, as more accurate decoders can tolerate higher physical error rates. To explore the optimal advantage of correlated decoding, we will compare decoding two surface codes jointly with MLE, and independently with MWPM [34, 35].

We study the performance of the transversal CNOT

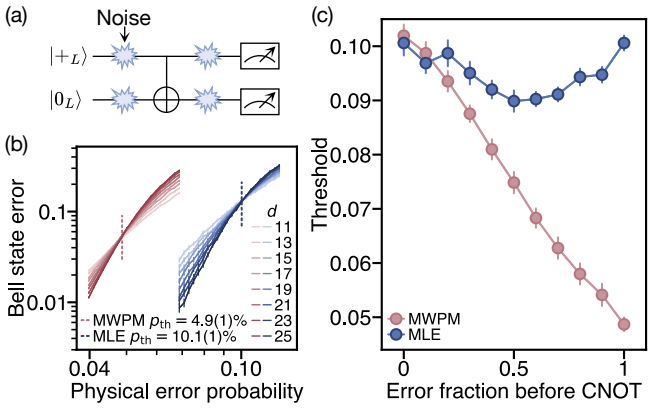

FIG. 2. Decoding the transversal CNOT. (a) We generate a logical Bell pair using a noisy transversal CNOT, with physical errors before and after the CNOT with probabilities pf_b and $p(1-f_b)$, respectively. (b) When the errors occur before the CNOT ($f_b = 1$), the threshold of uncorrelated MWPM (pink) is half that of MLE (blue). (c) As f_b increases, more errors are transferred between the logical qubits, and MLE increasingly outperforms MWPM.

by numerically generating a Bell pair between rotated surface codes, followed by noiseless syndrome extraction. Using Stim [27], we prepare the codes noiselessly in $|+_L 0_L\rangle$, then perform a CNOT with single-physical-qubit X and Z errors before and after with probabilities pf_b and $p(1-f_b)$, respectively (Fig. 2(a)). We use this simplified error model, parameterized by f_b, to modulate the fraction of errors transferred between qubits during the CNOT. Following the CNOT, the codes are first measured noiselessly in the X or Z basis to extract the syndrome, then decoded to compute the mean of the $X_L^1 X_L^2$ and $Z_L^1 Z_L^2$ populations as a proxy for fidelity [44]. Our decoding hypergraph includes vertices corresponding to the final stabilizer measurements, and inter- and intra-logical qubit hyperedges corresponding to the possible errors before and after the CNOT (see Fig. 1(b)) [32].

Because errors before the CNOT are copied between logical qubits, we expect the benefits of correlated decoding to grow with f_b. When the errors occur exclusively before the CNOT ($f_b = 1$), the density of errors on one logical qubit is doubled due to these copied errors. Therefore, the numerically-fitted [32] threshold of MWPM ($p_{\text{th}} = 4.9(1)\%$) is half that of MLE ($p_{\text{th}} = 10.1(1)\%$) (Fig. 2(b)). In Figure 2(c), we study the thresholds of MLE and MWPM as a function of f_b. When $f_b = 0$, no errors are transferred between logical qubits, so the thresholds of MWPM and MLE are equal.

Decoding deep logical Clifford circuits.— In realistic quantum algorithms, stabilizer measurements are used to remove entropy from the logical qubit by tracking physical errors over time. A challenge, however, is that the stabilizer measurements are also noisy. As a result, entangling gates based on lattice surgery require d rounds of syndrome extraction to ensure fault tolerance to these errors, creating considerable space-time overhead [45]. We now explore if we can perform fewer than d rounds between transversal CNOTs, by leveraging the deterministic propagation of stabilizer measurement errors through CNOTs to verify stabilizer measurements using surrounding rounds of syndrome extraction.

When constructing the decoding hypergraphs of circuits with noisy syndrome extraction, however, we observe that stabilizer measurement errors near transversal CNOTs generate hyperedges that make conventional decoding approaches challenging. To see this, consider two surface codes with syndrome extraction at times $t-1, t$, and $t+1$, with a transversal CNOT controlled on qubit 1 between times t and $t+1$. The checks associated with a particular Z stabilizer on both qubits are $\mathcal{Z}_{t-1}^1 \mathcal{Z}_t^1$ and $\mathcal{Z}_{t-1}^2 \mathcal{Z}_t^2$ ($\mathcal{Z}_t^1 \mathcal{Z}_{t+1}^1$ and $\mathcal{Z}_t^2 \mathcal{Z}_t^1 \mathcal{Z}_{t+1}^2$) for the first (last) two rounds, the latter of which comes from back-propagating the stabilizers through the CNOT. A measurement error on $\mathcal{Z}_t^1$ will thus flip three checks: $\mathcal{Z}_{t-1}^1 \mathcal{Z}_t^1$, $\mathcal{Z}_t^1 \mathcal{Z}_{t+1}^1$, and $\mathcal{Z}_t^2 \mathcal{Z}_t^1 \mathcal{Z}_{t+1}^2$ (Fig. 3(a)). Because this hyperedge has order three (meaning it connects three vertices) and cannot be decomposed into existing lower-order edges [32, 34], MWPM cannot be directly applied, as it only operates on edges of order ≤ 2. In the Supplementary Materials [32], we find that MWPM can be applied at the expense of d syndrome extraction rounds before a CNOT, but its threshold is substantially lower than that of the correlated decoders.

In contrast, because the correlated decoders can be applied to general decoding hypergraphs, they can potentially decode circuits with fewer than d syndrome extraction rounds between CNOTs. To study this, we numerically simulate deep logical Clifford circuits between four surface codes. The codes are initialized noiselessly in $|+_L\rangle^{\otimes 4}$, then entangled with 32 layers of transversal gates, with n_r rounds of syndrome extraction between layers (Fig. 3(b)). Each layer is comprised of a random transversal Hadamard H_L or Pauli gate $\{X_L, Y_L, Z_L\}$, followed by a random pairing of CNOTs with randomized designation of control and target. We add depolarizing noise to each circuit-level physical operation with probability p, as in Refs. [29, 32]. Finally, we noiselessly measure the X and Z stabilizers and the logical stabilizers of the resulting state, then decode to find the logical error rate P_L [46]. In the Supplementary Materials [32], we confirm that our results are insensitive to boundary effects from the final noiseless measurements, and representative of different randomly sampled circuits.

We first study the case of one round of syndrome extraction per CNOT ($n_r = 1$). Figure 3(c) shows the logical error rate per round, $P_{L,\max}\left[1-(1-P_L/P_{L,\max})^{1/(32n_r)}\right]$, where $P_{L,\max} = 1 - \frac{1}{2^4}$ is the error rate of the maximally mixed logical state, as a function of p [32]. The numerical results are consistent with the existence of a threshold

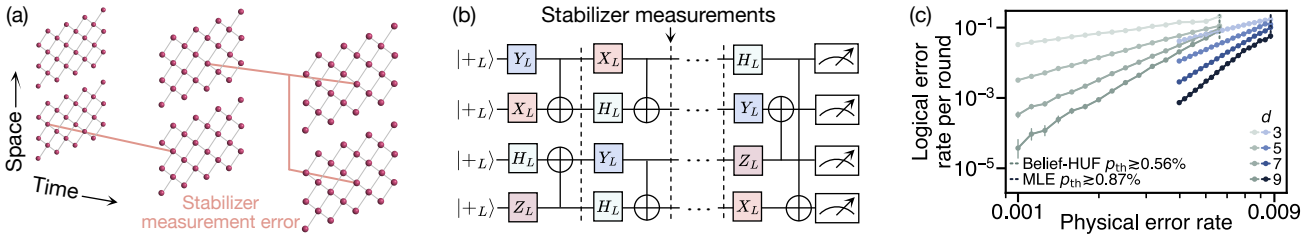

FIG. 3. Correlated decoding of deep logical circuits. (a) Stabilizer measurement errors near transversal CNOTs can appear as hyperedges connecting the checks of both logical qubits at different times. (b) We study deep logical Clifford circuits with n_r rounds of syndrome extraction between CNOTs. (c) Setting $n_r = 1$, our results are consistent with thresholds of $p_{\text{th}} \geq 0.56\%$ for belief-HUF (green) and $p_{\text{th}} \geq 0.87\%$ for MLE (blue).

for both belief-HUF and MLE. Though we cannot reliably fit the thresholds because $P_L/P_{L,\max} \to 1$ near the threshold, we estimate that $p_{\text{th}} \geq 0.56\%$ for belief-HUF and $p_{\text{th}} \geq 0.87\%$ for MLE from the largest physical error rate at which P_L decreases with d [32]. Furthermore, in the Supplementary Materials [32] we observe that the MLE scaling is consistent with achieving the full code distance.

Next, we investigate whether the space-time cost per CNOT, at a fixed physical error rate of $p = 0.001$, can be reduced by optimizing n_r. We express the space-time cost as $(n_r + 1)d^2$, corresponding to a cost of d^2 for the CNOT and $n_r d^2$ for syndrome extraction. In Figure 4(a), we plot P_L for various code distances and $n_r \in \{\frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1, \ldots, d\}$ ($n_r < 1$ corresponds to multiple CNOTs before a round of syndrome extraction). These results suggest that n_r can be reduced to $n_r \simeq \frac{1}{2}$ and $n_r \simeq 3$ for MLE and belief-HUF, respectively, without increasing P_L. P_L is expected to decrease exponentially with d according to the heuristic formula [25, 32]

$$P_L(n_r, d) = 32C n_r (\alpha_{n_r})^{\frac{d+1}{2}}. \tag{1}$$

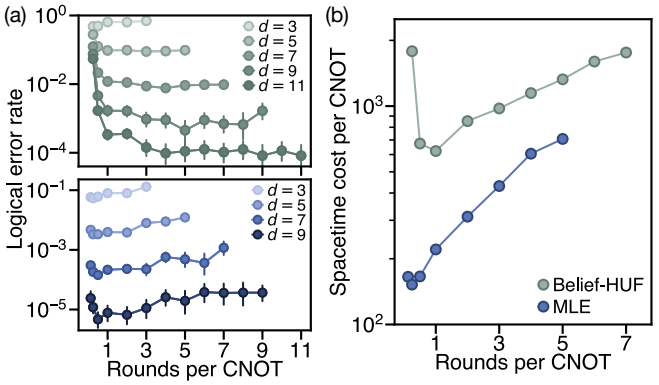

FIG. 4. Reducing the space-time cost of deep logical circuits. (a) The rounds of syndrome extraction per CNOT can be reduced to $n_r \simeq \frac{1}{2}$ for MLE (bottom) and $n_r \simeq 3$ for belief-HUF (top), without increasing P_L. (b) The extrapolated space-time cost to reach $P_L = 10^{-6}$ is minimized at $n_r = 1$ for belief-HUF and $n_r = \frac{1}{4}$ for MLE.

By fitting the coefficient C and exponent base α_{n_r}, we extrapolate the code distance needed to suppress P_L to 10^{-6}, from which we compute the space-time cost per CNOT (Fig. 4(b)). The space-time cost is minimized at $n_r = 1$ for belief-HUF and $n_r = \frac{1}{4}$ for MLE. Note that at $n_r = \frac{1}{4}$, the errors from the four transversal CNOTs balance the four physical CNOT layers from syndrome extraction [25, 32]. Therefore, the space-time cost of these logical circuits can be substantially reduced by using correlated decoding to optimize n_r.

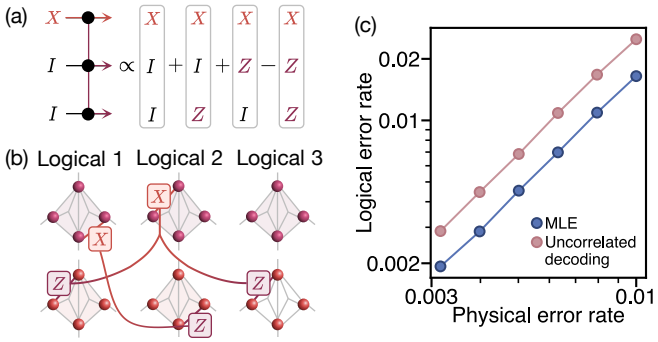

FIG. 5. Decoding the transversal CCZ. (a) A physical XII error before a CCZ propagates to $\frac{1}{2}(XII + XIZ + XZI - XZZ)$. (b) The decoding hypergraph includes a hyperedge for each Pauli error in the superposition. (c) The logical error rate is suppressed by a factor of $\simeq 1.5$ by decoding the logical qubits jointly.

Decoding transversal non-Clifford gates.— Correlated decoding can also be applied to non-Clifford transversal gates. As a proof-of-concept, we study a transversal CCZ between three fifteen-qubit quantum Hamming codes, with perfect syndrome extraction [32, 47]. We perfectly initialize each code in an X or Z-basis product state, then perform a CCZ with single-physical-qubit depolarizing noise beforehand. Because the CCZ is non-Clifford, these errors can propagate to a superposition of Pauli errors on multiple logical qubits, each generating a hyperedge in the decoding hypergraph (Fig. 5(a-b)) [32, 48]. We analyze initial states for which the circuit outputs a logical stabilizer state, which can be efficiently

simulated [32]. We then noiselessly measure the X and Z stabilizers and logical stabilizers, and decode to compute the average logical error rate over initial states [46]. Figure 5(c) shows that correlated decoding with MLE outperforms decoding the qubits independently by a factor of $\simeq 1.5$. Note that the gain here is smaller than with Clifford gates, as the error propagation is no longer fully deterministic. This motivates additional analysis of correlated decoding with non-Clifford gates as an avenue for future research.

Outlook.— Our results show that correlated decoding constitutes a powerful tool for improving the performance of logical algorithms. In particular, because stabilizer measurement errors can be tracked through the circuit, d rounds of syndrome extraction are not necessarily required between transversal CNOTs. These insights were used in the experiments of Ref. [17] to improve the entanglement fidelity of a logical Bell pair between surface codes, and to fault-tolerantly prepare logical GHZ states. Looking forward, these theoretical results indicate that the number of syndrome extraction rounds can be optimized for each particular circuit. In circuits where $\simeq 1$ round per CNOT is optimal, the space-time cost can be reduced by a factor of $\simeq d$.

While this technique already offers key performance advantages in the era of early fault-tolerant quantum computation, in the long term it has the potential to substantially reduce the space-time cost of large-scale algorithms [49, 50]. In particular, correlated decoding can be potentially applied to core algorithmic subroutines, such as magic-state distillation [24, 25] and quantum arithmetic [22, 26]. Moreover, the decoders we consider are readily applicable to other codes and gates, such as the recently developed quantum LDPC codes [51], space-time codes [28, 31], and the transversal S gate in the two-dimensional color code, whose decoding problem is challenging for standard decoders [52–54].

Our results can be extended in several directions. Although the computational runtime of belief-HUF is efficient, its threshold is lower than that of MLE on certain circuits (Fig. 3(c)). Alternative decoders, such as those utilizing tensor networks [55–57], belief propagation with ordered statistics decoding [58], or machine learning [59–61] could potentially close this performance gap. The runtime of belief-HUF can be further optimized by iteratively decoding fault-tolerant blocks rather than the full circuit [62–64]. These techniques can also be adapted to realistic experimental noise models, such as non-Pauli errors, biased noise [65], and erasure errors [66–68]. These considerations indicate that by co-designing the logical algorithm, error correcting code, and decoding strategy with experimental hardware in mind, the performance and resource requirements of logical algorithms can continue to be improved [65, 69].

Acknowledgements.— We thank C. Duckering for helpful suggestions on implementing the belief-HUF algorithm. We also thank S. Ebadi, S. Evered, S. Geim, A. Gu, M. Gullans, M. Kalinowski, A. Kubica, S. Li, T. Manovitz, N. Maskara, and C. Pattison for helpful discussions. We acknowledge financial support from IARPA and the Army Research Office, under the Entangled Logical Qubits program (Cooperative Agreement Number W911NF-23-2-0219), the DARPA ONISQ program (grant number W911NF2010021), the DARPA IMPAQT program (grant number HR0011-23-3-0012), the Center for Ultracold Atoms (a NSF Physics Frontiers Center, PHY-1734011), the National Science Foundation (grant number PHY-2012023 and grant number CCF-2313084), the NSF EAGER program (grant number CHE-2037687), the Army Research Office MURI (grant number W911NF-20-1-0082), the Army Research Office (award number W911NF2320219 and grant number W911NF-19-1-0302), and QuEra Computing. M.C. acknowledges support from Department of Energy Computational Science Graduate Fellowship under award number DE-SC0020347. J.P.B.A. acknowledges support from the Generation Q G2 fellowship and the Ramsay Centre for Western Civilisation. D.B. acknowledges support from the NSF Graduate Research Fellowship Program (grant DGE1745303) and The Fannie and John Hertz Foundation. This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

[1] P. W. Shor, Physical Review A **52**, R2493 (1995).
[2] J. Preskill, Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences **454**, 385 (1998).
[3] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Journal of Mathematical Physics **43**, 4452 (2002), arXiv:0110143 [quant-ph].
[4] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2010).
[5] A. Y. Kitaev, Annals of Physics **303**, 2 (2003).
[6] N. Ofek, A. Petrenko, R. Heeres, P. Reinhold, Z. Leghtas, B. Vlastakis, Y. Liu, L. Frunzio, S. M. Girvin, L. Jiang, M. Mirrahimi, M. H. Devoret, and R. J. Schoelkopf, Nature **536**, 441 (2016).
[7] P. Reinhold, S. Rosenblum, W.-L. Ma, L. Frunzio, L. Jiang, and R. J. Schoelkopf, Nature Physics **16**, 822 (2020).
[8] C. Ryan-Anderson, J. G. Bohnet, K. Lee, D. Gresh, A. Hankin, J. P. Gaebler, D. Francois, A. Chernoguzov, D. Lucchetti, N. C. Brown, T. M. Gatterman, S. K. Halit, K. Gilmore, J. A. Gerber, B. Neyenhuis, D. Hayes, and R. P. Stutz, Phys. Rev. X **11**, 041058 (2021).

[9] L. Postler, S. Heuβen, I. Pogorelov, M. Rispler, T. Feldker, M. Meth, C. D. Marciniak, R. Stricker, M. Ringbauer, R. Blatt, P. Schindler, M. Müller, and T. Monz, Nature **605**, 675 (2022).

[10] C. Ryan-Anderson, N. C. Brown, M. S. Allman, B. Arkin, G. Asa-Attuah, C. Baldwin, J. Berg, J. G. Bohnet, S. Braxton, N. Burdick, J. P. Campora, A. Chernoguzov, J. Esposito, B. Evans, D. Francois, *et al.*, Implementing fault-tolerant entangling gates on the five-qubit code and the color code (2022), arXiv:2208.01863 [quant-ph].

[11] G. Q. AI and Collaborators, Nature **618**, 264 (2023).

[12] G. Q. AI and Collaborators, Nature **614**, 676 (2023).

[13] D. H. Menendez, A. Ray, and M. Vasmer, Implementing fault-tolerant non-clifford gates using the [[8,3,2]] color code (2023), arXiv:2309.08663 [quant-ph].

[14] V. V. Sivak, A. Eickbusch, B. Royer, S. Singh, I. Tsioutsios, S. Ganjam, A. Miano, B. L. Brock, A. Z. Ding, L. Frunzio, S. M. Girvin, R. J. Schoelkopf, and M. H. Devoret, Nature **616**, 50 (2023).

[15] Y. Wang, S. Simsek, T. M. Gatterman, J. A. Gerber, K. Gilmore, D. Gresh, N. Hewitt, C. V. Horst, M. Matheny, T. Mengle, B. Neyenhuis, and B. Criger, Fault-tolerant one-bit addition with the smallest interesting colour code (2023), arXiv:2309.09893 [quant-ph].

[16] C. N. Self, M. Benedetti, and D. Amaro, Nature Physics 10.1038/s41567-023-02282-2 (2024).

[17] D. Bluvstein, S. J. Evered, A. A. Geim, S. H. Li, H. Zhou, T. Manovitz, S. Ebadi, M. Cain, M. Kalinowski, D. Hangleiter, J. P. Bonilla Ataides, N. Maskara, I. Cong, X. Gao, P. Sales Rodriguez, T. Karolyshyn, G. Semeghini, M. J. Gullans, M. Greiner, V. Vuletić, and M. D. Lukin, Nature **626**, 58 (2024).

[18] D. Bluvstein, H. Levine, G. Semeghini, T. T. Wang, S. Ebadi, M. Kalinowski, A. Keesling, N. Maskara, H. Pichler, M. Greiner, V. Vuletić, and M. D. Lukin, Nature **604**, 451 (2022).

[19] B. M. Terhal, Rev. Mod. Phys. **87**, 307 (2015).

[20] A. deMarti iOlius, P. Fuentes, R. Orús, P. M. Crespo, and J. E. Martinez, Decoding algorithms for surface codes (2023), arXiv:2307.14989 [quant-ph].

[21] D. Litinski, Quantum **3**, 128 (2019).

[22] C. Gidney and M. Ekerå, Quantum **5**, 433 (2021).

[23] D. Litinski and N. Nickerson, Active volume: An architecture for efficient fault-tolerant quantum computers with limited non-local connections (2022), arXiv:2211.15465 [quant-ph].

[24] S. Bravyi and A. Kitaev, Phys. Rev. A **71**, 022316 (2005).

[25] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Phys. Rev. A **86**, 032324 (2012).

[26] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, A new quantum ripple-carry addition circuit (2004), arXiv:quant-ph/0410184 [quant-ph].

[27] C. Gidney, Quantum **5**, 497 (2021).

[28] D. Gottesman, Opportunities and challenges in fault-tolerant quantum computation (2022), arXiv:2210.15844 [quant-ph].

[29] O. Higgott, T. C. Bohdanowicz, A. Kubica, S. T. Flammia, and E. T. Campbell, Phys. Rev. X **13**, 031007 (2023).

[30] M. McEwen, D. Bacon, and C. Gidney, Quantum **7**, 1172 (2023).

[31] N. Delfosse and A. Paetznick, Spacetime codes of Clifford circuits (2023), arXiv:2304.05943 [quant-ph].

[32] See Supplemental Material for (a) implementation details of the MLE and belief-HUF decoders, and (b) descriptions of numerical experiments, including preparing a Bell state with perfect syndrome measurements, a comparison of different decoders on preparing a Bell state with $2d$ noisy syndrome measurements, the deep logical transversal Clifford circuits, and the transversal CCZ with perfect syndrome measurements, which includes Refs. [70–77].

[33] D. Gottesman, An introduction to quantum error correction and fault-tolerant quantum computation (2009), arXiv:0904.2557 [quant-ph].

[34] O. Higgott, PyMatching: A Python package for decoding quantum codes with minimum-weight perfect matching (2021), arXiv:2105.13082 [quant-ph].

[35] O. Higgott and C. Gidney, Sparse blossom: correcting a million errors per core second with minimum-weight matching (2023), arXiv:2303.15933 [quant-ph].

[36] A. J. Landahl, J. T. Anderson, and P. R. Rice, Fault-tolerant quantum computing with color codes (2011), arXiv:1108.5738 [quant-ph].

[37] Y. Takada, Y. Takeuchi, and K. Fujii, Highly accurate decoder for topological color codes with simulated annealing (2023), arXiv:2303.01348 [quant-ph].

[38] Gurobi Optimization, LLC, Gurobi Optimizer Reference Manual (2023).

[39] E. Berlekamp, R. McEliece, and H. van Tilborg, IEEE Transactions on Information Theory **24**, 384 (1978).

[40] A. Vardy, IEEE Transactions on Information Theory **43**, 1757 (1997).

[41] N. Delfosse and N. H. Nickerson, Quantum **5**, 595 (2021).

[42] N. Delfosse, V. Londe, and M. E. Beverland, IEEE Transactions on Information Theory **68**, 3187 (2022).

[43] P. W. Shor, Fault-tolerant quantum computation (1997), arXiv:quant-ph/9605011 [quant-ph].

[44] G. Tóth and O. Gühne, Phys. Rev. A **72**, 022340 (2005).

[45] C. Horsman, A. G. Fowler, S. Devitt, and R. Van Meter, New Journal of Physics **14**, 123011 (2012), arXiv:1111.4022.

[46] D. Gottesman, The Heisenberg representation of quantum computers (1998), arXiv:quant-ph/9807006 [quant-ph].

[47] A. Paetznick and B. W. Reichardt, Phys. Rev. Lett. **111**, 090505 (2013).

[48] T. R. Scruby, M. Vasmer, and D. E. Browne, Phys. Rev. Res. **4**, 043052 (2022).

[49] P. W. Shor, SIAM Journal on Computing **26**, 1484 (1997).

[50] S. Lloyd, Science **273**, 1073 (1996).

[51] N. P. Breuckmann and J. N. Eberhardt, PRX Quantum **2**, 040101 (2021).

[52] P. Sarvepalli and R. Raussendorf, Phys. Rev. A **85**, 022317 (2012).

[53] N. Delfosse, Phys. Rev. A **89**, 012317 (2014).

[54] C. Gidney and C. Jones, New circuits and an open source decoder for the color code (2023).

[55] S. Bravyi, M. Suchara, and A. Vargo, Phys. Rev. A **90**, 032326 (2014).

[56] A. J. Ferris and D. Poulin, Physical Review Letters **113**, 030501 (2014).

[57] C. T. Chubb, General tensor network decoding of 2D Pauli codes (2021), arXiv:2101.04125 [quant-ph].

[58] P. Panteleev and G. Kalachev, Quantum **5**, 585 (2021).

[59] J. Bausch, A. W. Senior, F. J. H. Heras, T. Edlich, A. Davies, M. Newman, C. Jones, K. Satzinger, M. Y.

Niu, S. Blackwell, G. Holland, D. Kafri, J. Atalaya, C. Gidney, D. Hassabis, S. Boixo, H. Neven, and P. Kohli, Learning to decode the surface code with a recurrent, transformer-based neural network (2023), arXiv:2310.05900 [quant-ph].
[60] H. Cao, F. Pan, Y. Wang, and P. Zhang, qecgpt: decoding quantum error-correcting codes with generative pre-trained transformers (2023), arXiv:2307.09025 [quant-ph].
[61] H. Wang, P. Liu, K. Shao, D. Li, J. Gu, D. Z. Pan, Y. Ding, and S. Han, Transformer-qec: Quantum error correction code decoding with transferable transformers (2023), arXiv:2311.16082 [quant-ph].
[62] H. Bombín, C. Dawson, Y.-H. Liu, N. Nickerson, F. Pastawski, and S. Roberts, Modular decoding: parallelizable real-time decoding for quantum computers (2023), arXiv:2303.04846 [quant-ph].
[63] L. Skoric, D. E. Browne, K. M. Barnes, N. I. Gillespie, and E. T. Campbell, Parallel window decoding enables scalable fault tolerant quantum computation (2023), arXiv:2209.08552 [quant-ph].
[64] X. Tan, F. Zhang, R. Chao, Y. Shi, and J. Chen, PRX Quantum **4**, 040344 (2023).
[65] I. Cong, H. Levine, A. Keesling, D. Bluvstein, S.-T. Wang, and M. D. Lukin, Phys. Rev. X **12**, 021049 (2022).
[66] Y. Wu, S. Kolkowitz, S. Puri, and J. D. Thompson, Nature Communications **13**, 4657 (2022).
[67] S. Ma, G. Liu, P. Peng, B. Zhang, S. Jandura, J. Claes, A. P. Burgers, G. Pupillo, S. Puri, and J. D. Thompson, Nature **622**, 279 (2023).
[68] K. Sahay, J. Jin, J. Claes, J. D. Thompson, and S. Puri, Phys. Rev. X **13**, 041013 (2023).
[69] Q. Xu, J. P. B. Ataides, C. A. Pattison, N. Raveendran, D. Bluvstein, J. Wurtz, B. Vasic, M. D. Lukin, L. Jiang, and H. Zhou, Constant-overhead fault-tolerant quantum computation with reconfigurable atom arrays (2023), arXiv:2308.08648 [quant-ph].
[70] N. Delfosse and G. Zémor, Phys. Rev. Res. **2**, 033042 (2020).
[71] C. A. Pattison, M. E. Beverland, M. P. da Silva, and N. Delfosse, Improved quantum error correction using soft information (2021), arXiv:2107.13589 [quant-ph].
[72] D. Wiedemann, IEEE Transactions on Information Theory **32**, 54 (1986).
[73] D. Gottesman, Stabilizer codes and quantum error correction (1997), arXiv:quant-ph/9705052 [quant-ph].
[74] F. H. E. Watson and S. D. Barrett, New Journal of Physics **16**, 093045 (2014).
[75] M. E. Beverland, A. Kubica, and K. M. Svore, PRX Quantum **2**, 020341 (2021).
[76] C. Gidney, M. Newman, A. Fowler, and M. Broughton, Quantum **5**, 605 (2021).
[77] S. Aaronson and D. Gottesman, Physical Review A **70** (2004).
[78] N. Delfosse and J.-P. Tillich, in *2014 IEEE International Symposium on Information Theory* (IEEE, 2014).

Supplementary Materials

CONTENTS

1. DECODING ALGORITHMS

Here we describe the implementation details of the two decoding algorithms used to decode the transversal logical circuits considered in this work. Both decoders take in the check measurements and the decoding hypergraph, and return a physical error which could have generated the measured syndrome.

A. MLE decoder

The goal of the most-likely error (MLE) decoder is to exactly solve for the physical MLE consistent with the measured syndrome. We map the problem of finding the MLE onto a mixed-integer program, similar to the approaches in Refs. [36, 37]. Mixed-integer programs maximize a linear objective function over integer, binary, or real variables, subject to linear constraints on the variables. To solve for the MLE, we associate each potential physical error mechanism in the circuit (hyperedge in the decoding hypergraph) with a binary variable E_j that is equal to one if that error occurs, and zero otherwise. Our objective is then to maximize the total error probability, $\prod_{j=1}^{M} p_j^{E_j}(1-p_j)^{1-E_j}$, over the error variables $\{E_j\}_{j=1\ldots M}$. To make this objective function linear in the error variables, we will maximize the logarithm of the total error probability,

$$\log\left(\prod_{j=1}^{M} p_j^{E_j}(1-p_j)^{1-E_j}\right) = \sum_{j=1}^{M} \log(p_j)E_j + \log(1-p_j)(1-E_j). \tag{S1}$$

The total error is then constrained to be consistent with the measured syndrome. This means that if the ith check $C_i = +1$ (-1), then an even (odd) number of errors occurred in the set of errors which flip C_i, denoted $I(C_i)$. This constraint can be stated equivalently as $\sum_{E_j \in I(C_i)} E_j = \frac{1}{2}(1 + C_i) \mod 2$, which is almost linear, except for the modulo. To linearize the constraint, we introduce an integer slack variable K_i to the ith constraint which can add any multiple of 2 to the summation. The resulting mixed-integer program can be summarized as:

$$\begin{array}{lll} \text{maximize} & \sum_{j=1}^{M} \log(p_j)E_j + \log(1-p_j)(1-E_j) & \\ \text{subject to} & \sum_{E_j \in I(C_i)} E_j - 2K_i = \frac{1}{2}(1+C_i) & \forall i = 1,\ldots,N \\ & K_i \in \mathbb{Z}_{\geq 0} & \forall i = 1,\ldots,N \\ & E_j \in \{0,1\} & \forall j = 1,\ldots,M. \end{array}$$

To correct a given set of check measurements $\{C_i\}_{i=1\ldots N}$, we solve this mixed-integer program to optimality using Gurobi, a state-of-the-art solver [38], and apply the physical correction associated with the error indices j for which $E_j = 1$ in the optimal assignment.

B. Belief-HUF decoder

The original union-find (UF) decoder is a heuristic, almost-linear time surface code decoder which takes as input the *decoding Tanner graph* [41]. The decoding Tanner graph has vertices corresponding to both the checks and error mechanisms of the original decoding hypergraph, and an edge between each check vertex and error mechanism vertex which flips that check. UF operates in two main stages: first, it associates each Tanner graph vertex with a cluster, and grows each cluster along the edges of the Tanner graph, ceasing growth on a particular cluster once it is *satisfied*, meaning it contains a subset of error mechanisms which (if they occured) would generate a syndrome pattern consistent with the observed checks in that cluster. If two clusters meet, they merge into a single cluster. This process continues until each cluster is satisfied. In the second stage, UF applies the peeling decoder [70] to each cluster to find an error consistent with the measured checks, and applies a correction based on the predicted errors.

A key component of UF is the termination condition, which signifies that a cluster is satisfied. On the surface code, UF typically has two termination conditions: the total number of flipped checks in that cluster is even, or the cluster contains an error mechanism that flips only one check. Both of these conditions indicate that the cluster is satisfied, since errors in the bulk of the surface code only flip a pair of checks [25]. Note that the termination condition for a cluster can be evaluated in constant time by tracking the parity of the number of flipped checks, and the existence of an error mechanism in the cluster which flips one check.

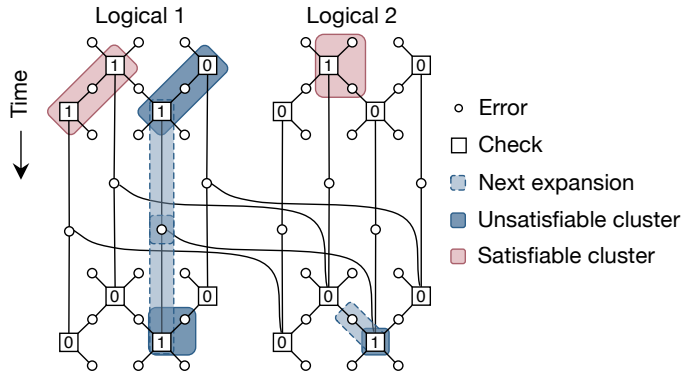

FIG. S1. Hypergraph union-find decoder. Clusters of vertices, each of which corresponds to a check or an error mechanism, are defined on the decoding Tanner graph. Each cluster expands along edges and merges with nearby clusters until it contains an error configuration consistent with its measured checks.

For logical circuits with transversal entangling gates, the presence of high-order errors (such as stabilizer measurement errors near CNOTs) requires the termination condition to be generalized (see Fig. 3(a), main text). Following Ref. [78], we replace this condition with the satisfiability of a linear system of equations, which directly checks whether a satisfying solution exists for the current cluster. A benefit of this approach is that solving the linear system of equations automatically provides the associated correction. As a trade-off, this termination condition increases the time complexity of the decoder by a polynomial overhead.

Concretely, the hypergraph union-find (HUF) decoder works by expanding clusters on the *decoding Tanner graph*. As illustrated in Figure S1, the decoding Tanner graph is a bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ constructed from the original decoding hypergraph. The vertices are comprised of two types, $\mathcal{V} = \{C_i\}_{i=1,\dots,M} \cup \{E_j\}_{j=1,\dots,N}$ where each C_i or E_j represents a check or an error mechanism, respectively, in the original decoding hypergraph. The edges $\mathcal{E}$ form a subset of $\{C_i\}_{i=1,\dots,M} \times \{E_j\}_{j=1,\dots,N}$, such that $(C_i, E_j) \in \mathcal{E}$ if and only if the edge E_j is incident to C_j in the original hypergraph, i.e., $E_j \in I(C_i)$. Edges are weighted according to the error probability of its connecting error mechanism:

$$w_{E_j,C_i} = \frac{\log\left(\frac{p_j}{1-p_j}\right)}{r_{E_j}^{\varepsilon}}, \tag{S2}$$

where r_{E_j} is the order of the hyperedge E_j (meaning the number of vertices it connects in the original decoding hypergraph), and ε is a hyperparameter of the HUF decoder. In Section 2 B, we compare the performance of HUF for different settings of ε.

Clusters are then defined as subsets of vertices in the decoding Tanner graph. An edge is called a *boundary edge* of a cluster if exactly one vertex in the edge is in that cluster. A vertex v is called *internal* to a cluster S, denoted as $v \in \bar{S}$, if all of its neighboring vertices belong to S. Given a sample of observed check measurements, a cluster is called *satisfiable* if there exists a configuration of internal error vertices consistent with the given observed checks in the cluster. Otherwise, it is called *unsatisfiable*.

The satisfiability of a cluster can be checked by a linear system solver over the binary field $\mathbb{F}_2$. Similar to the MLE algorithm in Section 1 A, the value of each error E_j can be regarded as an $\mathbb{F}_2$-valued variable, where $E_j = 1$ (0) indicates that the error E_j did (did not) occur. To be consistent with the observed check measurements, if the ith check $C_i = +1$ (-1), then an even (odd) number of errors occurred $I(C_i)$. This condition can be equivalently stated as

$$C_i = \prod_{E_j \in I(C_i)} (-1)^{E_j}. \tag{S3}$$

If we introduce σ_i such that $C_i = (-1)^{\sigma_i}$, then

$$\sigma_i = \sum_{E_j \in I(C_i)} E_j \qquad \mod 2. \tag{S4}$$

Consequently, σ_i can also be regarded as a variable in $\mathbb{F}_2$ that is linearly dependent on the error variables. In the rest this section, we will also refer to σ_i as a check variable for convenience. The satisfaction check algorithm is described in Algorithm 1.

Algorithm 1 Satisfaction check for a cluster

function CHECKSATISFACTION($\mathcal{G} = (\mathcal{V}, \mathcal{E}), \vec{\sigma}, S$)
  Initialize $\vec{E}|_S \leftarrow (0, \ldots, 0) \in \mathbb{F}_2^{|S|}$.
  Let C_S be all check vertices in S.
  Let $E_{\bar{S}}$ be all internal error vertices in S.
  Let H be a $|C_S| \times |E_{\bar{S}}|$ matrix over $\mathbb{F}_2$ such that

$$H_{C_i,E_j} = 1 \text{ iff } (C_i, E_j) \in \mathcal{E}.$$

  if $H\vec{E}|_{\bar{S}} = \vec{\sigma}|_S$ has a solution **then**
    Let $\vec{E}|_{\bar{S}}$ be a solution and update $\vec{E}|_S$.
    return (`satisfiable`, $\vec{E}|_S$)
  else
    return (`unsatisfiable`, $\vec{E}|_S$)
  end if
end function

As detailed in Algorithm 2, after assigning the value of the measured checks $\vec{\sigma} = (\sigma_1, \ldots, \sigma_N)$, the HUF decoder expands clusters on the decoding Tanner graph, prioritizing edges with lower weight w_{E_j,C_i} in order to search for a satisfiable configuration of errors within a cluster. The edge weights, and therefore the cluster growth, depend on the probability p_j of each error mechanism E_j, the number of checks r_{E_j} flipped by E_j, and the hyperparameter ε. Note that if one sets $\varepsilon = -1$ and applies Algorithm 2 to a decoding graph without hyperedges of order ≥ 3, then the expanding strategy reduces to the one defined on the split-edge graph proposed in Ref. [71].

The computational runtime of HUF grows polynomially with the size of the decoding Tanner graph M. In the worst case, one must merge all vertices into a single cluster to find a satisfiable error configuration, requiring $O(M + N) = O(M)$ total merge operations. By utilizing the union-find data structure [42], the complexity of each merge operation is only $O(\alpha(M+N))$, where α is the slowly-growing inverse Ackermann function. Each merge also requires a satisfaction check (Algorithm 1), which involves solving a linear system over $\mathbb{F}_2$. Naively, the linear system can be solved in $O(n^3)$ time using Gaussian elimination, where n is the number of variables in the system, corresponding to a worst-case total runtime of

$$O(M(M^3 + \alpha(M))) = O(M^4). \tag{S5}$$

However, in the circuits considered in this work, the linear system has constant sparsity, because each error mechanism flips a constant (in M) number of checks, and each check can be flipped by a constant number of error mechanisms. Therefore, the Wiedemann method can be used to solve the sparse linear system in $O(n^2 \log(n))$ time [72]. Consequently, the time complexity of the HUF decoder in this setting is at most

$$O(M(M^2 \log(M) + \alpha(M))) = O(M^3 \log(M)). \tag{S6}$$

Note that in practice, a merged cluster from the current round is unlikely to be updated in the next round, so the cluster update operations can be parallelized to improve the runtime. We leave further optimization of the HUF runtime as an avenue for future research.

The performance of HUF can be further augmented by using belief propagation (BP) before cluster expansion [29], yielding the belief-HUF algorithm we use in the main text. BP takes the decoding Tanner graph and the check measurements $\vec{\sigma}$ as input and, if converged, outputs the posterior probability of each error mechanism E_j, given $\vec{\sigma}$. Although BP is not guaranteed to converge because the decoding Tanner graphs of quantum codes have short loops, it can nevertheless be used to improve the performance of HUF. Following Ref. [29], we use the approximate posterior probabilities predicted by BP to update the error probabilities p_j used to weight the edges of the decoding Tanner graph in Eq. (S2). HUF then uses this updated decoding Tanner graph as input, expanding clusters at rates based on the updated probabilities. We use the sum-product message update rule, and execute a constant number of update rounds for the BP algorithm used in belief-HUF. We find in Section 2 B that belief-HUF significantly outperforms HUF in decoding the transversal CNOT with noisy syndrome extraction.

Algorithm 2 Hypergraph union-find decoder

function HYPERGRAPHUNIONFIND($\mathcal{G}, w, \vec{\sigma}$)
  Initialize a dictionary $f : \mathcal{E} \to \mathbb{R}$ to record the expansion state, such that $f(C_i, E_j) \leftarrow 0$
  Initialize a union-find data structure to maintain clusters of vertices in $\mathcal{G}$:

$$\mathcal{C} \leftarrow \{\{C_1\}, \ldots, \{C_M\}, \{E_1\}, \ldots, \{E_N\}\};$$

  Initialize a vector of unsatisfiable clusters as

$$\mathcal{C}_{\text{unsat}} \leftarrow [\{C_j\} \mid \sigma_j = 1,\ j = 1, \ldots, M];$$

  while $\mathcal{C}_{\text{unsat}} \neq \emptyset$ **do**
    Let $S \in \mathcal{C}_{\text{unsat}}$ be the cluster with the smallest size and least recently updated
    Delete S from $\mathcal{C}_{\text{unsat}}$
    Let B_S be the set of boundary edges of S
    Let $\Delta_f \leftarrow \min_{(C_i, E_j) \in S} w_{C_i, E_j} - f(C_i, E_j)$
    for $(C_i, E_j) \in B_S$ **do**
      $f(C_i, E_j) \leftarrow f(C_i, E_j) + \Delta_f$
    end for
    for $(C_i, E_j) \in B_S$ s.t. $f(C_i, E_j) = w_{C_i, E_j}$ **do**
      if $C_i \in S$ **then**
        Let S' be the cluster that contains E_j
      else
        Let S' be the cluster that contains C_i
      end if
      Let $S \leftarrow \text{MERGE}(\mathcal{C}, S, S')$
    end for
    $(R, \vec{E}|_S) \leftarrow \text{CHECKSATISFACTION}(\mathcal{G}, \vec{\sigma}, S)$
    if R is `unsatisfiable` **then**
      Add S into $\mathcal{C}_{\text{unsat}}$
    end if
  end while
  return A global error configuration $\vec{E} = \bigcup_{S \in \mathcal{C}} \vec{E}|_S$
end function

2. NUMERICAL SIMULATIONS

Here we describe the numerical simulations conducted to evaluate the performance of our correlated decoding algorithms. In general, in order to simulate a logical circuit, three components are specified:

- A *physical circuit* with *syndrome* measurements and *logical observable* measurements.
- A *noise model* applied to the physical circuit.
- A *decoder*.

The logical error rate is estimated by Monte-Carlo sampling the physical errors over the chosen noise model, then generating the syndrome and logical observable measurements corresponding to the sampled set of physical errors. The decoders then predict which errors occurred based on the syndrome measurements, and correct the logical observable based on this prediction. A logical error occurs if the corrected logical observable measurement differs from the true, noiseless value. In this work, we only simulate Clifford circuits with a noise model consisting of Pauli errors, with the exception of Section 2 D. These simulations can be performed in polynomial time in the number of physical qubits, according to the Gottesman-Knill theorem [73]. We use the open-source Clifford simulation package, Stim [27], to simulate the circuits studied in this work. Unless otherwise stated, the error bars for the numerically computed logical error rates are the Clopper-Pearson confidence interval based on a Beta distribution with a significance level of 0.05.

A. Surface code Bell state with perfect syndrome measurements

Here we give additional details for the numerical simulations in Figure 2 of the main text, which study preparing a logical Bell pair using a transversal CNOT. Using Stim [27], we first noiselessly prepare two rotated surface codes in

TABLE S1. Threshold comparison between different decoding algorithms.

Decoder	Hyperparameters	Decompose errors	Threshold
MWPM	—	`True`	$\simeq 0.49\%$
HUF 1	$\varepsilon = -1$	`False`	$\simeq 0.31\%$
HUF 2	$\varepsilon = 0$	`False`	$\simeq 0.59\%$
HUF 3	$\varepsilon = -1$	`True`	$\simeq 0.53\%$
HUF 4	$\varepsilon = 0$	`True`	$\simeq 0.78\%$
belief-HUF	$\varepsilon = 0$, `bp_rounds = 5`	`True`	$\simeq 0.95\%$
MLE	—	`False`	$\simeq 1.02\%$

$|+_L 0_L\rangle$, such that all stabilizers are initially +1. Next, we perform a noisy transversal CNOT by applying physical CNOT gates between the logical qubits. We insert single-physical-qubit X and Z errors with probability pf_b before the CNOT and $p(1-f_b)$ after the CNOT. Finally, we measure the physical qubits in the X or Z basis, allowing us to determine the stabilizer measurements and logical observables in that basis. We compare the MLE decoder described in Section 1 A against independently decoding the surface codes using the MWPM algorithm implemented in the open-source package PyMatching [34, 35].

As with the other logical circuits in this work, we use Stim to construct the decoding hypergraph. Because the codes are initialized perfectly, we define the checks simply as the stabilizers measured at the end of the circuit, as no information is gained from comparing the final stabilizer measurement with the +1 stabilizers during state preparation. These checks form the vertices of the hypergraph. The hyperedges correspond to the possible single-physical-qubit X and Z errors occuring before or after the CNOT. As illustrated in Fig. 1(b) of the main text, physical X (Z) errors before the CNOT on the control (target) qubit are transferred to the other logical qubit, in the bulk flipping four checks with probability pf_b, two for each code. Physical X (Z) errors before the CNOT on the target (control) qubit are not transferred to the other qubit, as well as errors after the CNOT. In the bulk, these errors only flip two checks on one surface code. Because MWPM cannot handle error sources that flip more than two checks, we decompose the hyperedges that flip four checks into two existing hyperedges that flip two checks within a code, using the `decompose_errors = True` hyperparameter in Stim. Because these edges already exist in the hypergraph, the error probabilities of the decomposed edges are combined with the existing error probabilities (see Ref. [27]).

To compare the performance of MLE and MWPM, we estimate the threshold p_{th} of both algorithms at different values of f_b. To estimate the logical error rate for each value of f_b, physical error rate p, and code distance d, we sample at least 40000 total samples or 6000 samples with logical errors, whichever condition is met first. For each f_b, we estimate the threshold by fitting the logical error rate P_L to the universal scaling hypothesis [74],

$$\log P_L = a + bx + cx^2, \tag{S7}$$

where $x = (p - p_{\text{th}})^{d^{1/\nu}}$. We fit the parameters $a, b, c, p_{\text{th}}, \nu$ using a least-squares fit for physical error rates $p = [p_{\text{th, guess}} - 0.017, p_{\text{th, guess}} + 0.027]$ and code distances $d \in \{17, 19, 21, 23, 25\}$, where $p_{\text{th, guess}}$ is an initial guess for the threshold which is close to the fitted value. The error bars plotted in Figure 2(c) of the main text are the 2σ error bars from the fit.

B. Surface code Bell state with $2d$ rounds of noisy syndrome measurements

Next, we consider using a transversal CNOT to prepare a Bell state between two rotated surface codes, with d rounds of noisy syndrome extraction before and after the CNOT, and circuit-level noise. We compare the performance of several decoders: MWPM (using PyMatching [34, 35]), MLE, belief-HUF, and HUF with several hyperparameter settings. In Table S1 and Figure S2(a) and (b), we show the approximate thresholds for each algorithm, estimated from the physical error rate at which the logical error rates of the two largest code distances intersect. For each algorithm, we obtain at least 10000 noisy samples per physical error rate and code distance.

To generate the circuit, we first initialize the rotated surface codes in $|+_L 0_L\rangle$. We initialize a code in $|0_L\rangle$ ($|+_L\rangle$) by initializing the physical qubits in $|0\rangle$ ($|+\rangle$), then measuring the X and Z stabilizers d times. During each round of stabilizer measurements, we first reset the ancilla qubits used to measure the X (Z) stabilizers in $|+\rangle$ ($|0\rangle$), then perform four CNOT gates between an ancilla qubit and its corresponding data qubits, using the same gate ordering and designation of control and target qubit as in Ref. [12], Fig. S12. Finally, we measure the ancilla qubits corresponding to the X (Z) stabilizers in the X (Z) basis. After state preparation, we perform a transversal CNOT controlled on the first qubit to generate a Bell pair, followed by d additional rounds of syndrome extraction, then a final projective measurement of the physical qubits in either the X or Z basis.

Throughout the circuit, we apply the circuit-level noise model in Ref. [29] to each physical operation. In this noise model, each physical two-qubit gate is followed by uniform two-qubit depolarizing noise with probability p. Qubits involved in a single-qubit gate or qubits which idle during a gate duration experience a single-physical-qubit depolarizing channel with probability p. Physical qubits initialized in the X (Z) basis are followed by a Z (X) error with probability $2p/3$, and measurements in the X (Z) basis are preceded by a Z (X) error with probability $2p/3$.

We decode using the $2d$ rounds of stabilizer measurements and the syndromes extracted from the final projective measurement. The checks of the decoding hypergraph are constructed from the product of each stabilizer measurement and the previous measurement of its backwards-propagated operator, if one exists. This means that a check comparing stabilizer measurements directly before and after the transversal CNOT can include stabilizer measurements from both the control and target logical qubit, as described in the main text. A check comparing two consecutive rounds of stabilizer measurements will simply be the product of those measurements. Using these checks, the decoding hypergraph is then constructed using Stim [27]. Finally, we compute the mean populations of $X_L^1 X_L^2$ and $Z_L^1 Z_L^2$ as a proxy for fidelity [44], as in Section 2 A.

1. *MWPM*

First, we discuss the decoding strategy of MWPM. As discussed in the main text, stabilizer measurement errors directly before the transversal CNOT generate order-three hyperedges in the decoding hypergraph. These hyperedges cannot be decomposed into a set of existing lower-order errors which together produce the same syndrome as the original hyperedge. Consequently, applying MWPM directly to the full decoding hypergraph will not yield a threshold, as MWPM cannot handle these order-three hyperedges. For example, if these hyperedges are removed from the decoding hypergraph in order to apply MWPM, then a perfect matching for a stabilizer measurement error directly before the CNOT (an order-three hyperedge) must connect to an order-one hyperedge on the boundary of the surface code, as errors in the bulk of the surface code only flip two checks. This procedure can erroneously apply a physical correction to the data qubits along the pairing, potentially creating a logical error.

In order to apply MWPM, we instead decode the circuit up to the CNOT, apply the associated correction, then decode and correct the circuit after the CNOT. To decode the first d syndrome extraction rounds before the CNOT, we associate higher-order hyperedges arising from stabilizer measurement errors directly before the CNOT with a time boundary in the decoding hypergraph. For example, a stabilizer measurement error directly before the CNOT flips a single check before the CNOT, and two checks after the CNOT (see main text). This generates an order-one time boundary edge in the decoding hypergraph before the CNOT, which can be handled by MWPM. We apply the correction associated with the edges in the matching, which resets the check measurements before the CNOT to +1. As a result of applying this correction, if the matching is paired to a stabilizer measurement error time boundary near the CNOT, we also flip the other checks after the CNOT that are affected by that error. We then decode and correct the final d rounds of syndrome extraction normally using MWPM.

To construct the decoding hypergraphs used to decode before and after the CNOT, we take the decoding hypergraph of the full circuit generated by Stim [27], and split it into two halves. The first half includes only checks whose stabilizer measurements occurred before the CNOT. Its hyperedges include all error mechanisms which only flip checks before the CNOT, as well as hyperedges connecting checks from both halves (including the order-three hyperedges from stabilizer measurement errors). These hyperedges on the boundary of the halves are filtered to only include their connections to checks before the CNOT. This filtering process may result in error sources which, when restricted to the checks before the CNOT, flip identical sets of checks. We merge these duplicate error sources into a single hyperedge with a combined probability $\sum_i \left(\frac{p_i}{1-p_i} \prod_j (1-p_j)\right)$ where p_i is the probability of each error source i in a group of duplicates. We use this modified hypergraph to decode before the CNOT, using the `decompose_errors=True` hyperparameter in Stim. To construct the decoding hypergraph after the CNOT, we only keep error mechanisms which flip checks solely after the CNOT. By using this hypergraph splitting strategy, MWPM achieves a threshold of $p_{\text{th}} \simeq 0.49\%$ (Table S1).

Note that this decoding approach requires d syndrome extraction rounds before the CNOT to be fault-tolerant to

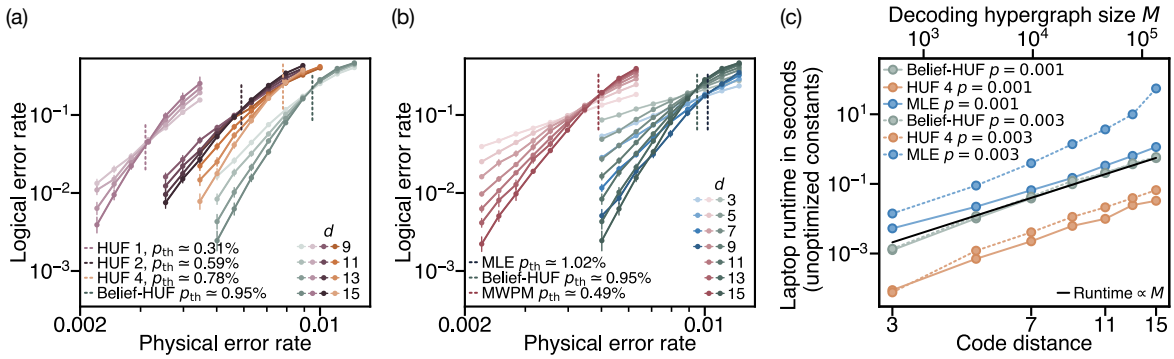

FIG. S2. Surface code Bell state with $2d$ rounds of noisy syndrome extraction. (a) The threshold of belief-HUF exceeds that of HUF 1 (light purple), HUF 2 (dark purple), and HUF 4 (orange). (b) Belief-HUF and MLE have similar thresholds, and both algorithms outperform MWPM. (c) The laptop runtimes of HUF 4 and belief-HUF grow polynomially with code distance for $p = 0.001$ (solid lines) and $p = 0.003$ (dotted lines), whereas the runtime of MLE appears to grow more rapidly for $p = 0.003$. The fitted laptop runtime of belief-HUF for $p = 0.001$ is approximately linear in the size of the decoding hypergraph M (black).

stabilizer measurement errors before the CNOT. As a result, this procedure is not fault-tolerant for logical algorithms with $\simeq 1$ syndrome extraction round between transversal CNOTs.

2. *Correlated decoding algorithms*

In contrast to MWPM, the correlated decoding algorithms we consider can utilize general decoding hypergraphs. Therefore, they can naturally handle the higher-order hyperedges arising from stabilizer measurement errors directly before a transversal CNOT. Here we benchmark four variants of HUF, which have different settings of the hyperparameter ε, which affects how quickly clusters grow along higher-order hyperedges (see Section 1 B). We also explore different settings of the `decompose_errors` parameter in Stim, which describes whether or not high-order hyperedges are decomposed into existing low-order hyperedges, resulting in an input decoding hypergraph with only essential hyperedges [27]. The thresholds of belief-HUF, MLE, and most variants of HUF exceed that of the MWPM (Table S1). Furthermore, the threshold of belief-HUF exceeds that of the HUF variants, and is comparable to that of MLE (Fig. S2(a) and (b)).

We also observe that the hyperparameter settings for HUF have a significant effect on performance (Table S1). For example, HUF 3 and HUF 4 gain a remarkable improvement over HUF 1 and HUF 2 by setting `decompose_error = True`. This is because although HUF can in principle handle higher-order hyperedges, decomposing the hyperedges restricts the rate of cluster expansion, as the size of a cluster increases more rapidly when expanding along high-order hyperedges compared to low-order ones. Concretely, as described in Section 1 B, an order-k hyperedge in the decoding hypergraph corresponds to k equally-weighted edges in the decoding Tanner graph. Therefore, the size of the cluster can increase by at most $k-1$ during one expansion step along this error mechanism. Excessively large clusters can result in decoding failures, as they involve solving a global linear system to find a satisfiable error, with no preference for choosing a high-probability error. Setting $\varepsilon = 0$ also decreases the priority by which the decoder expands clusters along hyperedges, further improving the performace of HUF 4 over HUF 3. Finally, the performance is further enhanced by introducing belief propagation with five rounds of belief propagation (`bp_rounds = 5`), yielding the belief-HUF algorithm. Note that both the transversal CNOT thresholds of HUF (0.78%) and belief-HUF (0.95%) with the setting of `decompose_errors = True` and $\varepsilon = 0$ match the thresholds of UF (0.795(1)%) and belief-find (0.937(2)%), respectively, in the surface code memory simulations reported in Ref. [29]. This confirms that the correlated decoders perform optimally compared to their uncorrelated versions, as we expect the memory threshold to be an upper bound on the threshold for the circuit we study with $2d$ rounds of syndrome extraction and a transversal CNOT.

Finally, we note that although the worst-case runtime of HUF derived in Section 1 B is $O(M^3 \log(M))$, where M is the size of the decoding hypergraph, in practice its runtime scaling is more favorable. In Figure S2(c), we plot the computational runtime per shot on a laptop for HUF, belief-HUF, and MLE for the physical error rates $p = 0.001$ and $p = 0.003$. A least-squares fit to the data at $p = 0.001$ to the functional form $y = ax^b$ yields a fit to the exponent parameter of $b = 1.00(3)$. Therefore, the runtime of HUF is linear in M at this physical error rate. We emphasize that the computational runtime of belief-HUF is not optimized for constant factors, and we anticipate that it can be

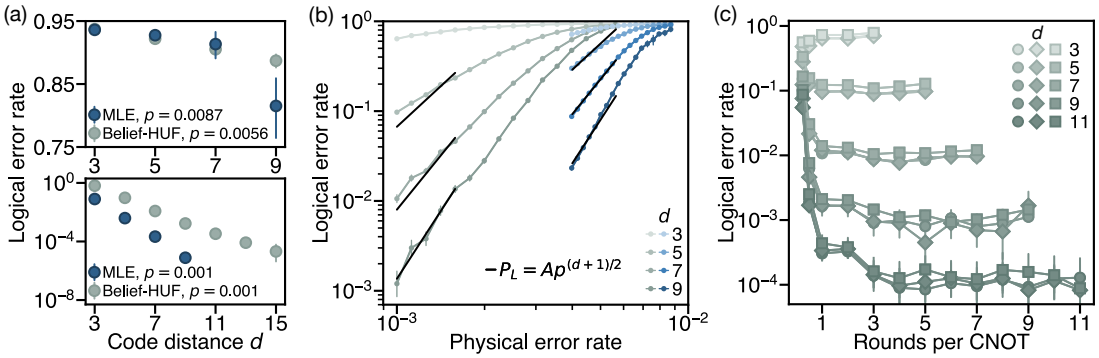

FIG. S3. Benchmarking deep logical circuits. (a) For the depth 32 logical circuit studied in the main text with $n_r = 1$ round of syndrome extraction per CNOT, the logical error rate decreases monotonically with code distance at physical error rates of $p = 0.87\%$ for MLE and $p = 0.56\%$ for belief-HUF (top). The same trend persists at a lower physical error rate of $p = 0.1\%$ (bottom). (a) We plot the logical error rate P_L as a function of the physical error rate p for the same circuit, and fit the coefficient A in the functional form $P_L = Ap^{(d+1)/2}$ to the data for the five smallest physical error rates studied (black lines). The quality of the fit suggests that MLE achieves the full code distance. (c) The average logical error rates over ten deep circuits with 32 (circles) and 40 (squares) layers of gates are similar as a function of n_r. Furthermore, the depth 32 instance studied in the main text (diamonds) has similar performance to the average over the depth 32 circuits.

substantially improved. We leave further optimization of the computational runtime as a subject for future research.

C. Deep logical transversal Clifford circuits

The deep logical Clifford circuits we study consist of four rotated surface code qubits and 32 layers of transversal gates, each consisting of single-qubit logic gates drawn from $\{H_L, X_L, Y_L, Z_L\}$, followed by a random pairing of transversal CNOTs with randomized designation of control and target. The transversal Pauli X_L, Y_L, and Z_L gates are performed by applying physical X, Y, and Z gates to all physical qubits within a code, respectively. The transversal Hadamard H_L gates are performed by applying H gates to all the qubits within a code, then noiselessly rotating the physical qubits by 90 degrees. After each layer of transversal CNOT gates, we perform n_r rounds of syndrome extraction, using the same circuit as in Section 2 B. Throughout the circuit, we apply the circuit-level noise model in Ref. [29] (see Section 2 B), except during state preparation, the final round of stabilizer measurements in the last transversal gate layer, the logical stabilizer measurements, and code rotation during transversal H_L gates, which are noiseless.

Figure 3(c) in the main text shows the logical error rate per round $P_{L,\text{round}}$ as a function of the physical error rate p. The logical error rate per round is given by the formula

$$P_{L,\text{round}} = P_{L,\max}\left[1 - (1 - P_L/P_{L,\max})^{1/(32n_r)}\right], \tag{S8}$$

where $P_{L,\max} = 1 - \frac{1}{2^4} = \frac{15}{16}$ is the error rate of the maximally mixed logical state. This formula can be derived by treating the process as a Markov chain, assuming that $P_{L,\text{round}}$ is a constant over different rounds, and that the four logical stabilizers each have an equal probability of flipping per round. If the input state to a round is the correct logical state, there is a probability of $1 - P_{L,\text{round}}$ of obtaining the correct output state. If the input state is an incorrect logical state, then the output state can be flipped back to the correct logical state with a probability of $P_{L,\text{round}}/15$. Note that when $P_{L,\text{round}}$ equals 0 ($P_{L,\max}$), P_L achieves its minimum (maximum).

Although our numerical results in Figure 3(c) of the main text are consistent with the existence of a threshold when $n_r = 1$, we cannot fit the exact threshold to the universal scaling hypothesis [74] because $P_L/P_{L,\max} \to 1$ at physical error rates near the threshold. Nevertheless, here we obtain a lower bound on the thresholds for belief-HUF and MLE by identifying the largest physical error rate consistent with a monotonic decrease in logical error rate per round with code distance. Figure S3(a) (top) demonstrates that the thresholds of belief-HUF and MLE are bounded by $p_{\text{th}} \geq 0.56\%$ and $p_{\text{th}} \geq 0.87\%$, respectively. Figure S3(a) (bottom) additionally shows that the logical error rate decreases exponentially for a larger range of code distances, at a lower physical error rate of $p = 0.1\%$.

In Figure S3(b), we show the logical error rate P_L as a function of the physical error rate p. We fit the coefficient A in the functional form $P_L = Ap^{(d+1)/2}$ for the smallest five physical error rates p studied for both belief-HUF and MLE

(black lines). This quality of the fits suggests that MLE achieves the full code distance, with exponential suppression of P_L in $(d+1)/2$. In contrast, the results for belief-HUF are less clear, and merit future research.

Next, we investigate whether the spacetime cost of the logical circuit needed to reach a fixed logical error rate can be reduced by optimizing n_r. To do so, we collect data at different values of n_r and d, displayed in Fig. 4(a) in the main text. The error bars in Figure 4(a), as with the rest of the text, correspond to the Clopper-Pearson confidence interval based on a Beta distribution with a significance level of 0.05. We use this data to fit the parameters C and α_{n_r} in the heuristic formula,

$$P_L(n_r, d) = 32 C n_r (\alpha_{n_r})^{\frac{d+1}{2}}, \tag{S9}$$

at different n_r. This formula comes from applying the standard formula for logical error rate as a function of code distance [25], modified so that the prefactor, $32Cn_r$, grows linearly with the circuit depth $32n_r$, similar to the numerical simulations in Ref. [75]. The fitted parameters for belief-HUF and MLE, obtained from a least-squares fit to the logarithm of Eq. (S9), are listed in Table S2. We only fit the exponent base α_{n_r} at values of n_r for which we have data for at least three different code distances, and for which the decoder is above the threshold at $p = 0.001$ (the latter condition is not fulfilled for belief-HUF at $n_r = 1/8$). Based on the fitted parameters, we can solve for the potentially fractional code distance needed to reach a target logical error rate of $P_L = 10^{-6}$ for each value of n_r. This gives us the extrapolated spacetime cost, expressed as $n_r d^2$, plotted in Fig. 4(b) in the main text. Note that the fitted exponent base α_{n_r} should not be interpreted as an accurate estimate of the ratio p/p_{th}, as previously observed in the literature [25, 76]. Additionally, we note that the fitted parameters may be subject to a small systematic errors due to the finite range of code distances studied. However, the resulting estimation error in the spacetime cost is likely small, as the target logical error rate $P_L = 10^{-6}$ is not significantly smaller than the minimum measured logical error rates $P_L \simeq 10^{-4}$.

Finally, we study whether our results are sensitive to finite size effects. In Figure S3(c), we compute the logical error rate as a function of n_r, averaged over ten randomly generated circuits with depths 32 and 40 (circles and squares, respectively). The results for both circuit depths are qualitatively similar, and quantitatively differ by an overall constant factor. This suggests that our results are not subject to finite-size effects due to the depth of the circuit. Furthermore, in Figure S3(c) we observe that the data for different random circuits at depth 32 is quantitatively similar to the particular depth 32 circuit studied in the main text (diamonds), indicating that our results are representative of other randomly generated circuits.

TABLE S2. Fit parameters for the heuristic scaling formula.

Parameter	**MLE**	**belief-HUF**
C	1.13(21)	0.464(65)
$\alpha_{\frac{1}{8}}$	0.097(7)	—
$\alpha_{\frac{1}{4}}$	0.070(5)	0.458(19)
$\alpha_{\frac{1}{2}}$	0.055(4)	0.240(10)
α_1	0.049(3)	0.170(7)
α_2	0.039(3)	0.146(6)
α_3	0.039(3)	0.120(5)
α_4	0.044(2)	0.109(4)
α_5	0.041(2)	0.102(4)
α_6	—	0.103(4)
α_7	—	0.097(4)

D. Transversal CCZ with perfect syndrome measurements

Here we study the performance of the transversal CCZ between three quantum Hamming codes, each comprised of fifteen physical qubits encoding seven logical qubits, with code distance three. This code is self-dual, meaning that the X and Z stabilizers are supported on the same physical qubits. The check matrix for each of the X and Z stabilizers is given by

$$H = \begin{pmatrix} 0&0&0&0&0&0&0&1&1&1&1&1&1&1&1 \\ 0&0&0&1&1&1&1&0&0&0&0&1&1&1&1 \\ 0&1&1&0&0&1&1&0&0&1&1&0&0&1&1 \\ 1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 \end{pmatrix}, \tag{S10}$$

where $H_{ij} = 1$ if qubit j is in the support of check i, and 0 otherwise. Each code block has seven logical qubits, six of which are gauge qubits, and one of which is used for the transversal CCZ. We work in the basis where the representation of the latter is transversal ($X_L = \bigotimes_{i=1}^{15} X_i$, $Z_L = \bigotimes_{i=1}^{15} Z_i$) [47].

The circuit we simulate prepares the three codes perfectly in an initial product state in the logical X or Z basis, then applies a single-physical-qubit depolarizing channel with probability p to the physical qubits, followed by a transversal CCZ. Because the transversal CCZ gate is non-Clifford, it cannot directly be simulated with a Clifford circuit via the Gottesman-Knill theorem [73]. Nevertheless, for particular initial states, the circuit can be efficiently simulated using a Clifford circuit, enabling us to probe the performance of the transversal CCZ on those initial states.

In particular, we efficiently simulate the circuit for initial product states for which the CCZ generates a known logical stabilizer state, $|\psi_L\rangle$. Some example initial states fulfilling this condition are $|1_L 1_L 1_L\rangle, |1_L 1_L +_L\rangle$, and $|1_L +_L +_L\rangle$, which after a CCZ generate the states $|1_L 1_L 1_L\rangle, |1_L 1_L -_L\rangle$, and $|1_L\rangle(|0_L +_L\rangle + |1_L -_L\rangle)/\sqrt{2}$, respectively. Notice that the output logical stabilizer state $|\psi_L\rangle$ can be efficiently constructed from a Clifford circuit using the Aaronson and Gottesman algorithm [77]. To simulate the noisy circuit efficiently, we therefore observe that introducing Pauli noise before the CCZ gate is equivalent to perfectly preparing $|\psi_L\rangle$, then subsequently applying the Clifford error obtained by propagating the Pauli error before the CCZ through the CCZ gate.

Therefore, we simulate the circuit by first generating $|\psi_L\rangle$ efficiently, then add the Clifford noise onto $|\psi_L\rangle$. To sample the Clifford error, we Monte-Carlo sample a random Pauli error before the CCZ drawn from a single-physical-qubit depolarizing channel, then propagate the sampled Pauli errors through the CCZ operation. In general, Z errors commute through the CCZ, and X (Y) errors propagate to an X (Y) error on the same qubit and a CZ error on the other two qubits in the CCZ. We then apply this propagated Clifford error onto the perfect state $|\psi_L\rangle$. Because the error itself is a Clifford gate, we can simulate the entire circuit efficiently [73] by repetitively sampling different error patterns.

We complete the circuit by noiselessly measuring the X and Z stabilizers and the logical stabilizer operators. The logical error rate for a given initial state is then computed from the corrected logical stabilizers [46], averaged over 20000 shots. To compute the total logical error rate, we average the logical error rate over five possible initial logical states: $|0_L 0_L 0_L\rangle, |0_L 0_L +_L\rangle, |0_L +_L +_L\rangle, |1_L 1_L +_L\rangle$, and $|1_L +_L +_L\rangle$. This choice comes from first symmetrizing over the three logical qubits, then using the fact that the $|0_L\rangle / |1_L\rangle$ and $|+_L\rangle / |-_L\rangle$ states will behave identically under noise.

To decode, we construct an approximate decoding hypergraph for the logical circuit. The decoding hypergraph is approximate because errors before the CCZ can propagate to a Clifford error after the CCZ [48], which can equivalently be written as a superposition of Pauli errors. Therefore, each error mechanism does not necessarily deterministically flip a particular set of checks. As an approximation, we therefore construct the decoding hypergraph to include a hyperedge for every Pauli error in the superposition. Concretely, for each group of three physical qubits undergoing a CCZ, we compute the input distribution of Pauli errors, e.g. the error XIZ occurs with probability $(p/3)^2(1-p)$. We then propagate each input Pauli error through the circuit, giving us a superposition of Pauli errors, e.g. XIZ goes to $\frac{1}{2}(XIZ + XII + XZZ - XIZ)$. We add a hyperedge to the decoding hypergraph corresponding to each term in the superposition. For example, the above error will generate hyperedges corresponding to the errors XIZ, XII, XZZ, and XIZ. To approximate the probability of each output error, we take the probability of the error before the CCZ, and normalize by the number of outputs (e.g., each of the above errors is associated with a probability of $(p/3)^2(1-p)/4$). If two different Pauli error sources propagate to the same error (e.g., the errors XII and XIZ will both propagate to a superposition of Pauli errors which includes XZI), then we sum the resulting error probabilities in the decoding hypergraph.

To compare correlated decoding against independently decoding the logical qubits, we construct two different decoding hypergraphs: one which includes all error sources, as described above, and one which includes only error

sources acting with a single logical qubit. For correlated decoding, we use the MLE algorithm (Section 1 A) with the former decoding hypergraph as input. To independently decode the qubits, we use MLE with the latter decoding hypergraph as input.