Fault-tolerant one-bit addition with the smallest interesting color code

Wang, Yang; Simsek, Selwyn; Gatterman, Thomas M.; Gerber, Justin A.; Gilmore, Kevin; Gresh, Dan; Hewitt, Nathan; Horst, Chandler V.; Criger, Ben; More Authors

DOI
10.1126/sciadv.ado9024

Publication date
2024

Document Version
Final published version

Published in
Science Advances

Citation (APA)
Wang, Y., Simsek, S., Gatterman, T. M., Gerber, J. A., Gilmore, K., Gresh, D., Hewitt, N., Horst, C. V., Criger, B., & More Authors (2024). Fault-tolerant one-bit addition with the smallest interesting color code. *Science Advances*, *10*(29), Article eado9024. https://doi.org/10.1126/sciadv.ado9024

APPLIED SCIENCES AND ENGINEERING

Fault-tolerant one-bit addition with the smallest interesting color code

Yang Wang[1,2], Selwyn Simsek[3], Thomas M. Gatterman[4], Justin A. Gerber[4], Kevin Gilmore[4], Dan Gresh[4], Nathan Hewitt[4], Chandler V. Horst[4], Mitchell Matheny[4], Tanner Mengle[4], Brian Neyenhuis[4], Ben Criger[3,5]*

Fault-tolerant operations based on stabilizer codes are the state of the art in suppressing error rates in quantum computations. Most such codes do not permit a straightforward implementation of non-Clifford logical operations, which are necessary to define a universal gate set. As a result, implementations of these operations must use either error-correcting codes with more complicated error correction procedures or gate teleportation and magic states, which are prepared at the logical level, increasing overhead to a degree that precludes near-term implementation. Here, we implement a small quantum algorithm, one-qubit addition, fault-tolerantly on a trapped-ion quantum computer, using the [[8, 3, 2]] color code. By removing unnecessary error correction circuits and using low-overhead techniques for fault-tolerant preparation and measurement, we reduce the number of error-prone two-qubit gates and measurements to 36. We observe arithmetic errors with a rate of $\sim 1.1 \times 10^{-3}$ for the fault-tolerant circuit and $\sim 9.5 \times 10^{-3}$ for the unencoded circuit.

INTRODUCTION

Quantum computers have a large and growing number of potential applications (*1*), and quantum computers of increasing size are being constructed (*2*, *3*).

Owing to the effects of noise and physical imperfections, these devices continue to have large physical error rates, on the order of 2×10^{-3} (*4*) for a typical two-qubit entangling gate or measurement, preventing the direct implementation of large-scale algorithms with physical qubits. Therefore, it is necessary to carry out a quantum computation fault-tolerantly to reduce the effective error rate.

One straightforward way to construct a fault-tolerant circuit is to select a quantum error-correcting code together with a corresponding set of fault-tolerant operations, and replace the individual operations of a given non–fault-tolerant circuit with their fault-tolerant counterparts (*5*). Error correction gadgets designed for the code in question are then inserted between every adjacent pair of fault-tolerant operations to prevent the effect of errors building up over time. Proofs of the threshold theorem often provide an explicit construction of such a procedure (*6*). However, this procedure often involves a large overhead in terms of both gate count and number of qubits required, which limits the suitability of such circuits to be implemented on experimental hardware.

Reducing the size of a fault-tolerant circuit (in terms of gate and qubit count) can make implementation easier. Smaller circuits are also preferable as there is reason to expect that they result in lower logical error rates. For instance, consider two fault-tolerant circuits C_s and C_l that act identically on all inputs, where the circuit C_s is shorter than the circuit C_l, and both circuits can detect or correct $t - 1$ faults. C_l contains more locations at which faults may occur and, therefore, has a higher likelihood of experiencing at least t faults, which may be undetectable/uncorrectable and contribute to the logical error rate, provided all other factors are equal. This is another reason why smaller fault-tolerant circuits are to be preferred over larger ones.

Much previous work employs a conventional approach to implementing fault-tolerant algorithms. One starts with a code in which it is straightforward to implement Clifford gates directly. For instance, this can be done with surface code patches via lattice surgery (*7*), and we note that an instance of Grover's algorithm not containing non-Clifford gates has already been implemented fault-tolerantly on two qubits (*8*) using the [[4, 2, 2]] code. Then, to obtain a universal gate set, a single non-Clifford gate such as the T gate is implemented by gate teleportation (*9*), which may require magic state distillation (*10*).

One may instead start with codes that have transversal (and hence fault-tolerant) non-Clifford gates. The Clifford gates are then implemented by gate teleportation with Pauli eigenstates as the ancillary inputs, requiring no expensive distillation to prepare.

Here, we realize a small instance of this proposal. We implement a one-qubit addition circuit using the [[8, 3, 2]] color code, which allows a transversal non-Clifford CCZ gate. Although this algorithm is simple, it computes the answer to a mathematical problem and contains both Clifford and non-Clifford gates. The one-bit adder circuit is also a simple application of the Toffoli gate, which is a universal gate for reversible classical computing. It can be used to construct oracles for Grover's algorithm (*8*, *11*) as well as the modular exponentiation circuits used within Shor's algorithm. As a result, the fault-tolerant realization of a one-bit adder circuit has practical implications for the realization of more substantial quantum algorithms, as well as pushing forward the state of the art in the realization of fault-tolerant algorithms on contemporary quantum computers.

RESULTS

Experimental details

Using the techniques outlined in Materials and Methods, we are able to write a fault-tolerant implementation of one-bit addition in the [[8, 3, 2]] color code using 24 CNOTs and 12 measurements (single-qubit operations are not counted for the purpose of estimating the final logical error rate, due to their much lower error rates) (see Fig. 1D).

[1]QuTech, Delft University of Technology, PO Box 5046, 2600 GA Delft, Netherlands. [2]3. Physikalisches Institut, ZAQuant University of Stuttgart, Allmandring 13, 70569 Stuttgart, Germany. [3]Quantinuum Terrington House, 13–15 Hills Road, Cambridge CB2 1NL, UK. [4]Quantinuum 303 South Technology Ct., Broomfield, CO 80021, USA. [5]Institute for Globally Distributed Open Research and Education (IGDORE).
*Corresponding author. Email: ben.criger@quantinuum.com

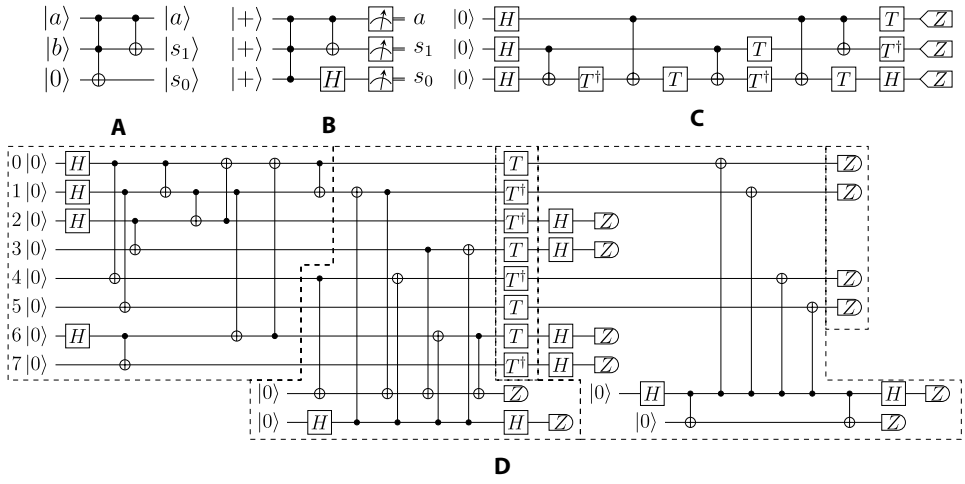

Fig. 1. Quantum circuits that realize addition of two one-qubit numbers. (**A**) One-bit addition circuit. The result of $a + b$ is stored in the two-bit number $s = s_0 s_1$. (**B**) One-bit addition with the uniform superposition as input, preparing the "superposition of valid sums" and measuring it destructively. (**C**) Non–fault-tolerant circuit as submitted to compiler. (**D**) Fault-tolerant implementation of one-bit addition given in Fig. 1B as submitted to the compiler. Dashed regions, from left to right: non–fault-tolerant $|\overline{+++}\rangle$ preparation, flag fault-tolerant measurement of $\overline{X}_1\overline{X}_3$ and overlapping S_Z, transversal CCZ, destructive measurement of $\overline{X}_2$, and destructive measurement of $\overline{Z}_1$ and $\overline{Z}_3$.

For comparison, decomposing the one-qubit adder into CNOTs and single-qubit gates on bare qubits results in five CNOTs and three measurements (see Fig. 1C). These were submitted to both the Quantinuum H1-1 quantum computer and the Quantinuum H1-1E emulator. Upon submission, a compiler transforms each circuit into one corresponding to the native gate set of the quantum computer. The resulting circuits were executed 10,000 times on the quantum computer and 100,000 times on the emulator. Instances in which the fault-tolerant circuit returned any nonzero syndrome or flag outcome were rejected, returning 8998 instances from the device and 88,537 from the emulator with no errors detected. Frequencies at which the four incorrect outputs occur due to undetected errors are presented in Fig. 2.

To completely characterize the logical errors that occur in a fault-tolerant one-qubit addition, it would be necessary to perform three-qubit logical process tomography. However, process tomography results in prohibitively high sampling overhead and introduces the challenge of distinguishing state preparation and measurement (SPAM) errors from those occurring in the unitary of interest. For these reasons, we instead execute a complete protocol consisting of state preparation, unitary operations, and measurements and calculate an operationally defined error rate that is affected by all steps of the process. At the end of each summation, we measure the register at the physical level and calculate classical values for the one-bit number a and the two-bit number s. If

$$(s = 3) \vee ((s = 2) \wedge (a = 0)) \vee ((s = 0) \wedge (a = 1)) \qquad (1)$$

we can infer that a logical error has occurred. We call these events arithmetic errors and compare the rates at which they occur in Fig. 2.

To evaluate the effectiveness of the arithmetic error rate as a measure of the true error rate, we attempt to estimate the true error rate. This cannot be done directly from the data presented in Fig. 2. Therefore, we carry out a density matrix–based simulation in QuTiP (*12*) [see (*13*) for source code] of the circuit, using a noise model motivated by that of the Quantinuum H1-1E emulator

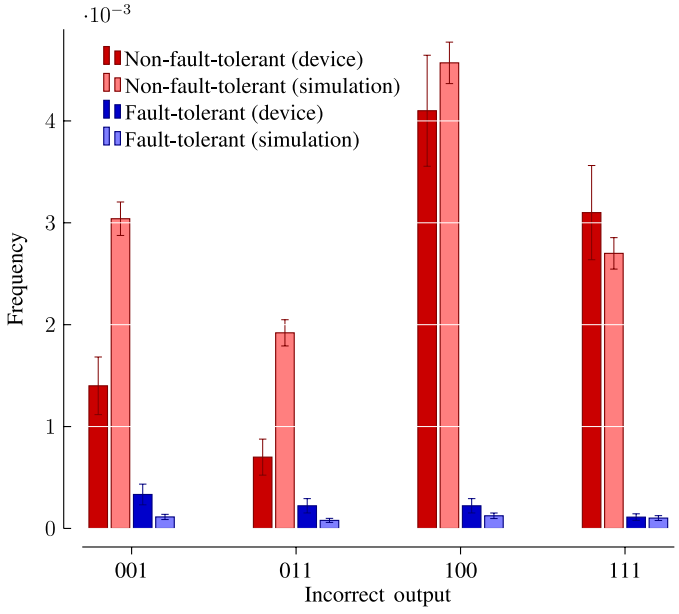

Fig. 2. Results of evaluating fault-tolerant and non–fault-tolerant circuits on both the Quantinuum H1-1 quantum computer and emulator. Only those shots that pass postselection are used to estimate the arithmetic error rate of the fault-tolerant circuit. The overall rates of arithmetic errors for the non–fault-tolerant circuit are 0.95 ± 0.19% (on H1-1) and 1.22 ± 0.07% (on the emulator); for the fault-tolerant circuit, they are 0.11 ± 0.07% (on H1-1) and 0.04 ± 0.014% (on the emulator).

(which is in turn constructed using experimental data from the H1-1 device), and attempt to quantify the effect of noise on the state obtained at the end of the circuit. We obtain an arithmetic error rate of 0.39%, which is of comparable magnitude to that observed in Fig. 2. The fidelity of the simulated state with the ideal error-free state is 99.7%, which implies that the use of the

arithmetic error rate does not conceal significant logical errors in other bases.

Comparison with planar architectures

To understand the overhead reduction achieved in this implementation, we investigate a surface code–based implementation of one-bit addition for surface code patches with distance $d = 2$. Similar to (*14*), logical qubits are realized in independent code blocks, logical Clifford gates are implemented using lattice surgery, and the logical Hadamard is propagated forward, resulting in an $\overline{X}$ measurement on the third qubit at the end of the circuit. We can lower the overhead of this implementation by generating the state $CCZ\,|\overline{+++}\rangle$ using a magic state factory (*15*), and computing with it directly, rather than using it for gate teleportation. At distance $d = 2$, the factory can detect any single-qubit error during the production of the $|\mathrm{CCZ}\rangle$ state.

The overhead of this implementation is dominated by the CCZ magic state factory, whose measurement schedule is shown in Fig. 3. It requires 18 surface code patches to implement, which at distance $d = 2$ implies $\sim 18 \cdot 2d^2 = 144$ physical qubits. This figure is over an order of magnitude higher than that of the [[8, 3, 2]] code implementation discussed above and is too large to be executed on a Quantinuum H-series device at time of writing.

The relative logical error rates resulting from different implementations of one-bit addition can be estimated by counting pairs of faults that result in logical errors (larger sets of independent faults being much less likely). We carry this estimation out using QuantumClifford.jl (*16*, *17*), resulting in 84,873 fault pairs for the $d = 2$ surface code, and 1116 for the [[8, 3, 2]] code. While a more detailed comparison would not be applicable to devices of either architecture, we can see that the number of malicious pairs is far greater than the number of malicious pairs for the [[8, 3, 2]] code–based implementation, which suggests that higher code distance (and greater overhead) would be necessary to match the logical error rate obtained in this work.

By contrast, implementing one-qubit addition using the [[8, 3, 2]] color code on a device with square-lattice connectivity can be accomplished with moderate overhead using the qubit layouts in Fig. 4A. The initial layout is used for non–fault-tolerant state preparation, and four CNOTs are then required to change the layout so that the remainder of the experiment can be carried out. After non–fault-tolerant state preparation (shown in Fig. 4B), the stabilizer measurement requires an additional six CNOTs, since the SWAP gates must be decomposed into CNOTs rather than replaced with transport operations. While these additional CNOTs represent a significant increase in the overall size of the circuit without increasing its ability to tolerate errors, the induced overhead is not as significant as implementing the computation with surface code–based logical qubits.

DISCUSSION

The fault-tolerant implementation of one-bit addition demonstrates the combined effect of transversal non-Clifford gates, logical Cliffords by permutation, postselected state preparation, and omission of superfluous error correction gadgets (such as those after transversal single-qubit or automorphism gates) in a fault-tolerant computation. While we cannot expect each of these techniques to result in the same logical error rate reduction in all fault-tolerant computations, we believe that each of them will contribute to lower logical error rates and overheads in some future fault-tolerant computations. This result also highlights the peculiar "inversion of difficulty" in fault-tolerant quantum computing. That is, the operations that induce the most error at the physical level (CNOT and CCZ) can be

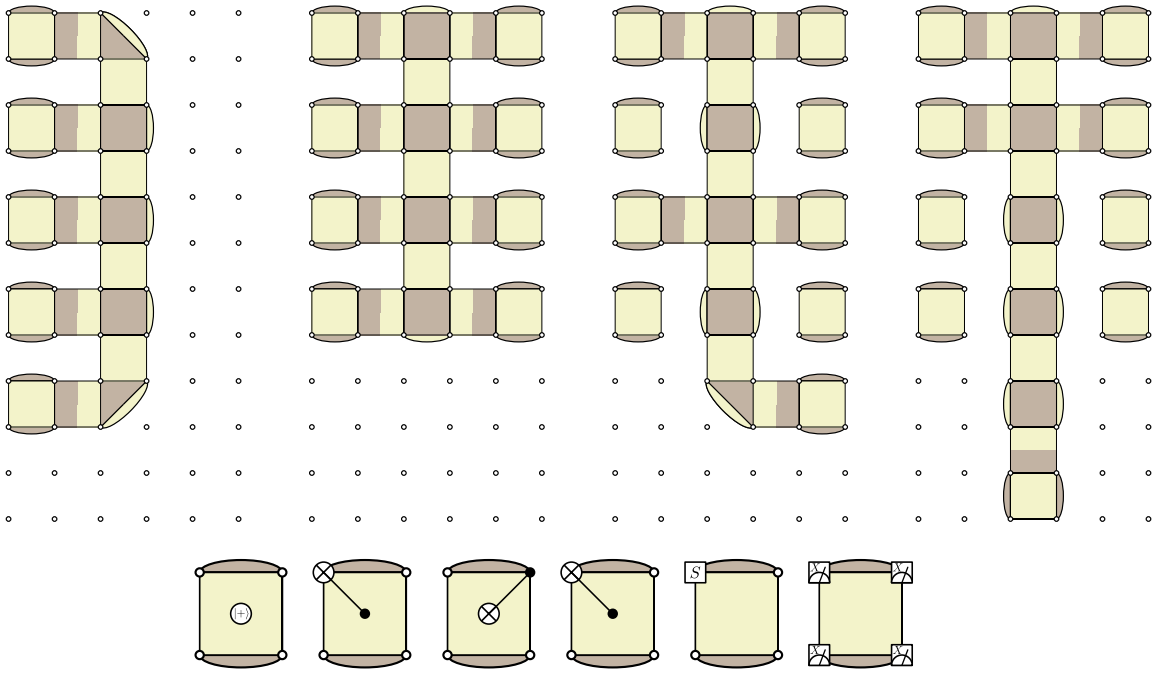

Fig. 3. Schedule for simulating a CCZ state factory using lattice surgery with distance-2 surface codes. Top: Multi-qubit logical $\overline{X}$ measurements performed in series. Bottom: Destructive distance-1 measurement in the $\overline{SX}$ basis, applied to the eight surface codes used as logical ancillas. Note that, in a genuine magic state factory (*15*), T and $T^\dagger$ gates would take the place of the S gates used here, which we use here for ease of simulation.

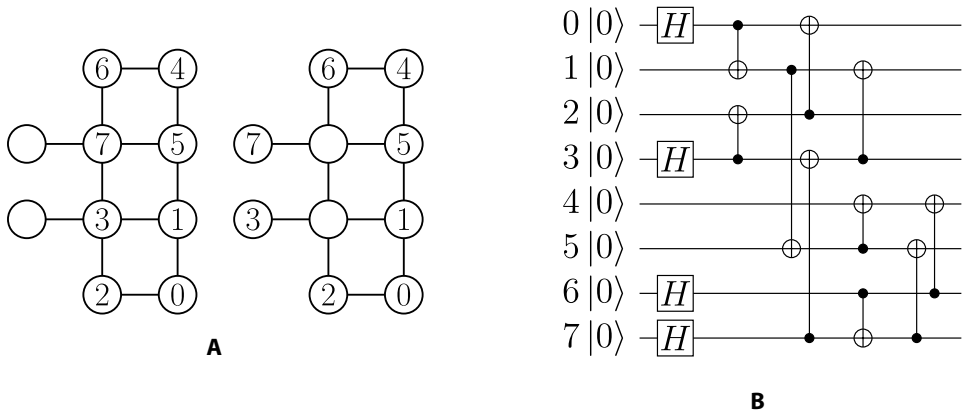

Fig. 4. Fault-tolerant one-bit addition on a planar architecture. (**A**) Small planar qubit layouts that facilitate fault-tolerant one-bit addition. Left: Layout for non–fault-tolerant state preparation. Right: Layout for subsequent flag-based measurement of $X_{1,2,5,6}/Z_{1,2,5,6}$ and $X_{1,3,5,7}$. (**B**) Modified non–fault-tolerant $|\overline{+++}\rangle$ preparation circuit, using the minimal number of CNOTs and qubits connected in a square lattice.

carried out fault-tolerantly using high-fidelity transport and transversal single-qubit gates. In contrast, state preparation (which is comparatively simple and reliable at the physical level) comprises most of the fault-tolerant circuit, causing a large fraction of logical errors.

A further example of this inversion of difficulty is the logical Hadamard gate, which is required to generate a universal gate set and cannot be implemented transversally. One possibility is to implement this gate by teleportation, involving the preparation of an ancillary Pauli eigenstate of the [[8, 3, 2]] code. While this technique would require lower overhead than magic state distillation, the need for an ancillary encoded state doubles the number of data qubits necessary to complete a universal gate set. In "Logical Hadamard gates on the [[8, 3, 2]] code" in the Supplementary Materials, we derive a lower-overhead alternative, taking advantage of the transversal CNOT between the [[8, 3, 2]] code and the [[4, 2, 2]] code shown in Fig. 5D.

The fault-tolerant implementation of one-bit addition also highlights several open problems to address in future work. For instance, the experiment considered here uses postselection rather than correction and achieves a low conditional probability of error with moderate (~10%) postselection overhead. Full postselection (i.e., accepting the output only if no syndrome is observed) on every fault-tolerant gadget would result in an exponentially decaying acceptance probability. Still, the effect of partial postselection (accepting syndromes that indicate an error with weight $w \ll d/2$ on selected gadgets within a larger algorithm) has yet to be explored.

In addition, the placement of quantum error correction (QEC) gadgets between every adjacent pair of fault-tolerant operations is sufficient to prove the existence of thresholds in the ExRec formalism (*6*). However, it is not necessary to do this to make a circuit fault-tolerant. Omitting QEC gadgets (or using partial QEC gadgets, as in "Logical Hadamard gates on the [[8, 3, 2]] code" in the Supplementary Materials) between consecutive fault-tolerant operations can reduce logical error rates and overheads in a wide variety of protocols.

Finally, we note that the fault-tolerant protocol we have developed leverages variable connectivity of the physical qubits and exhibits relatively high pseudothresholds for certain quantum computing tasks, making it particularly suitable for other platforms with high qubit connectivity, such as neutral atoms (*18*, *19*) and nitrogen-vacancy (NV)–based networks (*20*, *21*). A similar idea has recently been proposed to implement quantum low-density parity-check codes (qLDPCs) using reconfigurable atom arrays (*18*). By using the product structure inherent in many qLDPC codes to implement nonlocal syndrome extraction circuits via atom rearrangement, effectively constant overhead in practically relevant regimes can be achieved (*18*).

MATERIALS AND METHODS

One-bit addition

At the logical level, two bits (potentially in superposition) may be added using the circuit in Fig. 1A. Note that a classical one-bit addition requires only two bits of memory, as two bits are sufficient to store both the input bits and the output, which may be a two-bit number. However, the resulting circuit is not reversible, as the input bits cannot be recovered from their sum. To implement the one-bit adder on a quantum computer, a reversible classical circuit is needed. The one-bit adder can be made reversible at the cost of requiring three bits or qubits.

Implementing the circuit in Fig. 1A with computational basis states as inputs would not be a good demonstration of quantum computing, as the state would not exhibit superposition or entanglement throughout the computation. To remedy this, we input the state $|+\rangle|+\rangle$, thus preparing the equal superposition of all four two-bit sums and obtaining the result of one of them at measurement time. We also replace the Toffoli gate with a CCZ gate conjugated by Hadamards, which, after simplification, gives the circuit of Fig. 1B, which is the logical circuit we use in the remainder of this work. In the following section, we review the error-detecting code selected for this work (the [[8, 3, 2]] color code). We express the logical circuit in terms of simple, low-overhead fault-tolerant operations in the "Circuit construction" section.

[[8, 3, 2]] color code

To encode the three logical qubits necessary for one-bit addition and gain access to a transversal CCZ, we select the [[8, 3, 2]] color code (*22*, *23*) (see Fig. 5A for a description of the stabilizers and logical Pauli operators). To prepare a uniform superposition of valid one-bit

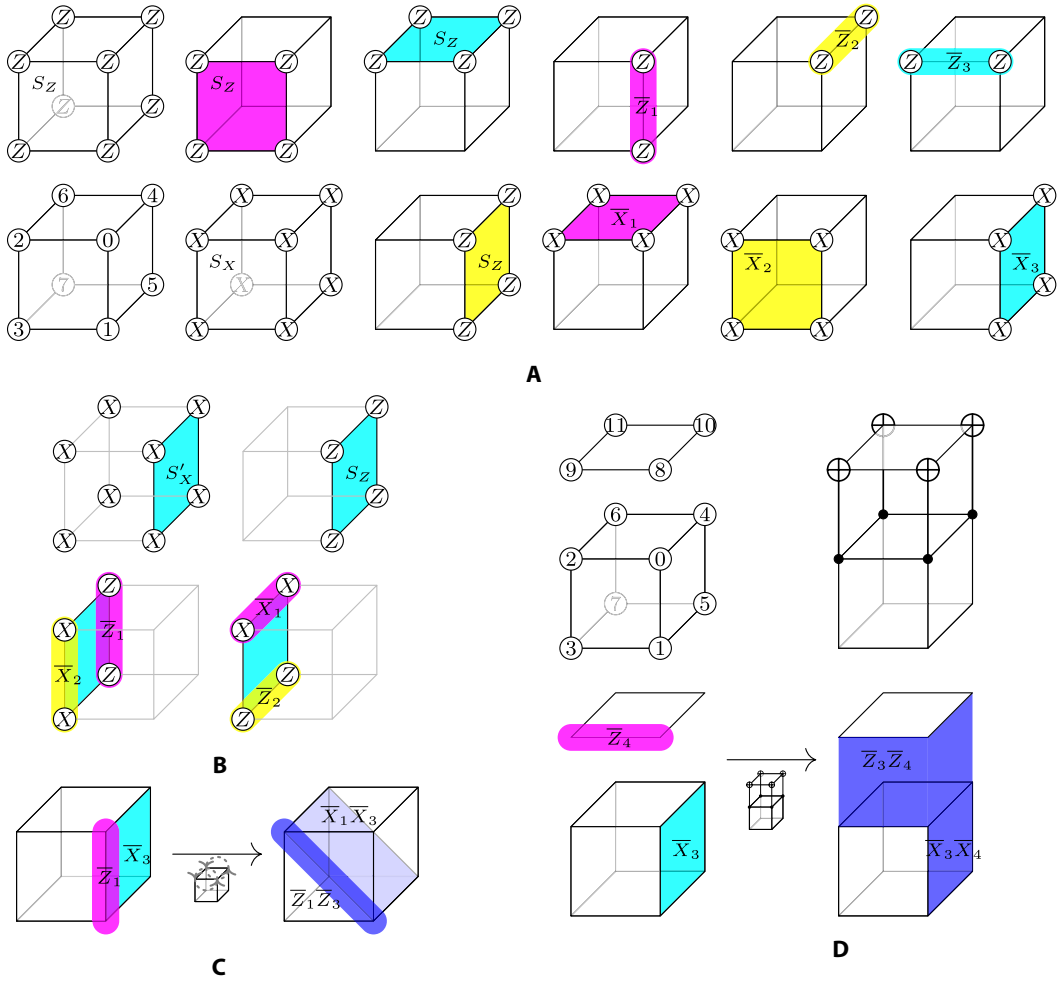

Fig. 5. Stabilizers and logical operations of the [[8, 3, 2]] color code. (**A**) Stabilizer generators and logical operators for the [[8, 3, 2]] color code. (**B**) Action of measuring every qubit of the left face of the cube in the *X* basis. (**C**) Action of reflecting the top face of the cubic layout for the [[8, 3, 2]] code. The operators $\overline{X}_2$ and $\overline{Z}_3$ are mapped to $\overline{X}_2\overline{X}_3$ and $\overline{Z}_2\overline{Z}_3$, respectively, leaving other logical operators unaffected, effectively performing a logical CNOT gate. (**D**) Transversal logical CNOT between a face of the [[8, 3, 2]] and [[4, 2, 2]] codes. $\overline{Z}$ operators contained in the specified face and $\overline{X}$ operators that share an edge with that face are affected.

sums, we first fault-tolerantly prepare the $|\overline{+}\rangle^{\otimes 3}$ state. We then perform the transversal CCZ, and a CNOT between qubits 1 and 2, then measure qubit 3 in the $\overline{X}$ basis, and finally measure qubits 1 and 2 in the $\overline{Z}$ basis. While the transversal CCZ of the [[8, 3, 2]] code is already well understood, the state preparation, Clifford gates, and measurements used here have not been described in the previous literature to our knowledge. We explain their derivation below, in order of execution in the experiment (shown in Fig. 1D).

Circuit construction

$|\overline{+}\rangle^{\otimes 3}$ preparation

There is a well-known procedure for fault-tolerant preparation of $|\overline{+}\rangle^{\otimes k}$ states in CSS codes of distance d involving measuring Z stabilizers with the $|+\rangle^{\otimes n}$ state as input. The first round of measurement will result in random outcomes, so multiple rounds (two for codes with $d = 2$) would be necessary to detect measurement errors. For the [[8, 3, 2]] code, this requires ~32 CNOTs and 8 measurements.

To reduce the size of this circuit, we use the Goto circuit design technique (*24*), first writing out a non–fault-tolerant circuit with the minimum number of two-qubit gates and then measuring a limited set of stabilizers that detect high-weight errors resulting from error propagation through the initial circuit. We find the non–fault-tolerant stage of the circuit by inspection, beginning from the desired final state and using CNOT gates to break the state's entanglement, until arriving at a state consisting of four Bell pairs, which can be prepared fault-tolerantly using CNOTs on bare qubits. We confirm that this circuit contains the minimum number of CNOTs using breadth-first search over an implicit graph whose vertices are canonical stabilizer states [see (*16*, *25*, *26*)].

Gottesman-Knill simulation reveals that all high-weight propagated errors can be detected by fault-tolerant measurement of two weight-four stabilizers, $X_{1,3,4,6}$ and $Z_{1,3,4,6}$ (note: the only high-weight errors not detectable by later stabilizer measurement are $Z_{0,1}$ and $Z_{0,6}$; had this been used as the criterion for fault tolerance, a flag-based measurement with two fewer CNOTs could have been used). This can be accomplished using an appropriately interleaved circuit designed by Reichardt (*27*). The final preparation circuit requires 18 CNOTs and two measurements, halving the number of relatively error-prone gates with respect to the generic technique.

Destructive $\overline{X}$ measurement

Similarly to state preparation, there is also a generic protocol for measuring the logical observables of CSS codes. In this protocol, all data qubits are measured in the X or Z basis, and the eigenvalues of stabilizers and logical operators in that basis are then reconstructed by calculating classical parities, allowing a final round of classical error correction to be performed. In this way, any set composed only of tensor products of $\overline{Z}$ or $\overline{X}$ operators can be measured, but we cannot simultaneously measure $\overline{X}_j$ and $\overline{Z}_{k\neq j}$ using this protocol.

Typically, whenever we wish to measure a subset of logical qubits in a different basis, we would use a fault-tolerant circuit with additional ancillas to measure the relevant operators nondestructively (similarly to stabilizer measurements) or synthesize the logical measurement by transferring the relevant subset of logical qubits to a new code block, which may subsequently be measured destructively. The $[[8, 3, 2]]$ color code allows an alternative to such a protocol; a logical X operator supported on a face may be measured destructively by measuring the four qubits on that face in the X basis. The remaining $\overline{X}$ operators are reduced to weight two, and the weights of the $\overline{Z}$ operators not supported on the measured face are preserved, leaving the remaining two logical qubits encoded in the $[[4, 2, 2]]$ code (see Fig. 5B).

This destructive measurement is not fault-tolerant because no stabilizer eigenvalue can be reconstructed from the measurement outputs. To reconstruct $S_X = X^{\otimes 8}$, we use a flag-based circuit (see Fig. 1D, right-hand side) to nondestructively measure $X^{\otimes 4}$ on the opposite face and take the parity of the five measurement outputs to reconstruct the stabilizer eigenvalue.

Logical CNOT by permutation

With CSS codes that encode a single logical qubit, such as surface code patches, entangling operations are implemented across different code blocks, usually using transversal gates or lattice surgery (*28*). For codes such as the $[[8, 3, 2]]$ code, which have $k > 1$ logical qubits, there is no general protocol for performing logical entangling operations within a single code block [though architectures that use codes with $k > 1$ have been explored (*29*)].

However, the $[[8, 3, 2]]$ color code has a logical CNOT gate, which can be implemented by permuting or relabeling the physical qubits. This is due to a spatial symmetry of the stabilizer group that the logical operators do not obey. The QCCD architecture (*2*, *30*) allows us to transport physical qubits from one area of a device to another if necessary. As a result, the CNOT gate given in Fig. 1B can be implemented with near-unit fidelity using only transport operations (see Fig. 5C).

Supplementary Materials

This PDF file includes:

Supplementary Text
Fig. S1
References

REFERENCES AND NOTES

1. A. Montanaro, Quantum algorithms: An overview. *npj Quantum Inf.* **2**, 15023 (2016).
2. J. M. Pino, J. M. Dreiling, C. Figgatt, J. P. Gaebler, S. A. Moses, M. S. Allman, C. H. Baldwin, M. Foss-Feig, D. Hayes, K. Mayer, C. Ryan-Anderson, B. Neyenhuis, Demonstration of the trapped-ion quantum CCD computer architecture. *Nature* **592**, 209–213 (2021).
3. IBM Newsroom, IBM unveils breakthrough 127-qubit quantum processor (2021); https://newsroom.ibm.com/2021-11-16-IBM-Unveils-Breakthrough-127-Qubit-Quantum-Processor.
4. Quantinuum, Quantinuum system model H1 product data sheet version 5.20 (2022); https://www.quantinuum.com/hardware/h1.
5. P. W. Shor, Fault-tolerant quantum computation, in *Proceedings of 37th Conference on Foundations of Computer Science* (IEEE, 1996), pp. 56–65.
6. P. Aliferis, D. Gottesman, J. Preskill, Quantum accuracy threshold for concatenated distance-3 code. *Quantum Inf. Comput.* **6**, 97–165 (2006).
7. D. Litinski, F. von Oppen, Lattice surgery with a twist: Simplifying Clifford gates of surface codes. *Quantum* **2**, 62 (2018).
8. B. Pokharel, D. Lidar, Better-than-classical Grover search via quantum error detection and suppression. *npj Quantum Inf.* **10**, 23 (2024).
9. D. Gottesman, I. L. Chuang, Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. *Nature* **402**, 390–393 (1999).
10. S. Bravyi, A. Kitaev, Universal quantum computation with ideal Clifford gates and noisy ancillas. *Phys. Rev. A* **71**, 022316 (2005).
11. L. K. Grover, A fast quantum mechanical algorithm for database search, in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (ACM, 1996), pp. 212–219.
12. J. R. Johansson, P. D. Nation, F. Nori, QuTiP 2: A Python framework for the dynamics of open quantum systems. *Comput. Phys. Commun.* **184**, 4, 1234–1240 (2013).
13. S. Simsek, B. Criger, One bit addition density matrix simulation. Commit 1daca27 (2024); https://github.com/CQCL/One_Bit_Addition_Density_Matrix_Simulation.
14. D. Litinski, A game of surface codes: Large-scale quantum computing with lattice surgery. *Quantum* **3**, 128 (2019).
15. C. Gidney, A. G. Fowler, Efficient magic state factories with a catalyzed $|CCZ\rangle$ to $2\,|T\rangle$ transformation. *Quantum* **3**, 135 (2019).
16. Quantum Savory, QuantumClifford.jl (2023); https://github.com/QuantumSavory/QuantumClifford.jl.
17. B. Criger, LatticeSurgeryCCZStateExample.jl. Commit fdb8784 (2023); https://github.com/CQCL/LatticeSurgeryCCZStateExample.jl.
18. Q. Xu, J. P. B. Ataides, C. A. Pattison, N. Raveendran, D. Bluvstein, J. Wurtz, B. Vasić, M. D. Lukin, L. Jiang, H. Zhou, Constant-overhead fault-tolerant quantum computation with reconfigurable atom arrays. *Nat. Phys.* **20**, 1084–1090 (2024).
19. S. J. Evered, D. Bluvstein, M. Kalinowski, S. Ebadi, T. Manovitz, H. Zhou, S. H. Li, A. A. Geim, T. T. Wang, N. Maskara, H. Levine, G. Semeghini, M. Greiner, V. Vuletić, M. D. Lukin, High-fidelity parallel entangling gates on a neutral-atom quantum computer. *Nature* **622**, 268–272 (2023).
20. Y. Wang, "Using spins in diamond for quantum technologies," thesis, Delft University of Technology, The Netherlands (2023).
21. S. L. N. Hermans, M. Pompili, H. K. C. Beukers, S. Baier, J. Borregaard, R. Hanson, Qubit teleportation between non-neighbouring nodes in a quantum network. *Nature* **605**, 663–668 (2022).
22. A. Kubica, B. Yoshida, F. Pastawski, Unfolding the color code. *New J. Phys.* **17**, 083026 (2015).
23. E. T. Campbell, The smallest interesting colour code (2016); https://earltcampbell.com/2016/09/26/the-smallest-interesting-colour-code/.
24. H. Goto, Minimizing resource overheads for fault-tolerant preparation of encoded states of the Steane code. *Sci. Rep.* **6**, 19578 (2016).
25. E. Scheinerman, ImplicitGraphs. Commit 14459d3 (2022); https://github.com/scheinerman/ImplicitGraphs.jl.
26. B. Criger, P.-J. Derks, S. Sivarajah, Clifford BFS. Commit 3007ced (2023); https://github.com/CQCL/Clifford_BFS.
27. B. W. Reichardt, Fault-tolerant quantum error correction for Steane's seven-qubit color code with few or no extra qubits. *Quantum Sci. Technol.* **6**, 015007 (2021).
28. B. M. Terhal, Quantum error correction for quantum memories. *Rev. Mod. Phys.* **87**, 307–346 (2015).
29. T. A. Brun, Y.-C. Zheng, K.-C. Hsu, J. Job, C.-Y. Lai, Teleportation-based fault-tolerant quantum computation in multi-qubit large block codes. arXiv:1504.03913 [quant-ph] (2015).
30. D. Kielpinski, C. Monroe, D. J. Wineland, Architecture for a large-scale ion-trap quantum computer. *Nature* **417**, 709–711 (2002).
31. Y. Shi, Both Toffoli and controlled-NOT need little help to universal quantum computing. *Quantum Inf. Comput.* **3**, 84–92 (2003).
32. B. Eastin, E. Knill, Restrictions on transversal encoded quantum gate sets. *Phys. Rev. Lett.* **102**, 110502 (2009).
33. D. Horsman, A. G. Fowler, S. Devitt, R. Van Meter, Surface code quantum computing by lattice surgery. *New J. Phys.* **14**, 123011 (2012).
34. M. B. Hastings, J. Haah, Distillation with sublogarithmic overhead. *Phys. Rev. Lett.* **120**, 050504 (2018).
35. E. Knill, Quantum computing with realistically noisy devices. *Nature* **434**, 39–44 (2005).

Acknowledgments: We thank A. Landahl for inspiring this project, P.-J. Derks for pointing out a flaw in a previous version of the software used to check fault tolerance, as well as R. Duncan,

C. Self, C. Ryan-Anderson, and D. Hayes for helpful comments and insightful discussions during the preparation of this work, and K. Mayer for illuminating discussions regarding the noise model used on the Quantinuum H1-1 emulator. We also thank D. H. Menendez, A. Ray, and M. Vasmer for fruitful discussions. **Funding:** This publication is part of the QuTech NWO funding 2020-2024—Part I "Fundamental Research" with project number 601.QT.001-1, which is financed by the Dutch Research Council (NWO). This work was supported by the German Federal Ministry of Education and Research (BMBF) through the project QECHQS with grant no. 16KIS1590K. **Author contributions:** B.C. conceived the original idea. Y.W. and B.C. developed the circuits. S.S. ran the circuits and obtained experimental data. B.C. counted the number of fault pairs that result in logical errors in the surface code implementation. Y.W. and B.C. devised the single and double logical Hadamard gates described in the Supplementary Materials. T.M.G., J.A.G., K.G., D.G., N.H., C.V.H., M.M., T.M., and B.N. operated, optimized, and maintained the Quantinuum H1-1 quantum computer from which experimental data were obtained. B.C. and S.S. took the lead in writing the manuscript. **Competing interests:** B.C. and S.S. hold employment-related stock options in Quantinuum and/or its group companies. This declaration is made to ensure transparency and does not imply any conflict of interest. The remaining authors declare no competing interests. **Data and materials availability:** All data needed to evaluate the conclusions in the paper are present in the paper and/or the Supplementary Materials. Experimental data are available at doi:10.5281/zenodo.11282005.

Submitted 26 February 2024
Accepted 13 June 2024
Published 19 July 2024
10.1126/sciadv.ado9024