nature physics

Article

https://doi.org/10.1038/s41567-024-02479-z

Constant-overhead fault-tolerant quantum computation with reconfigurable atom arrays

Received: 16 August 2023

Accepted: 15 March 2024

Published online: 29 April 2024

Check for updates

Qian Xu[1,6], J. Pablo Bonilla Ataides[2,6], Christopher A. Pattison[3], Nithin Raveendran[4], Dolev Bluvstein[2], Jonathan Wurtz[5], Bane Vasić[4], Mikhail D. Lukin[2], Liang Jiang[1] ✉ & Hengyun Zhou[2,5] ✉

Quantum low-density parity-check (qLDPC) codes can achieve high encoding rates and good code distance scaling, potentially enabling low-overhead fault-tolerant quantum computing. However, implementing qLDPC codes involves nonlocal operations that require long-range connectivity between qubits. This makes their physical realization challenging in comparison to geometrically local codes, such as the surface code. Here we propose a hardware-efficient scheme for fault-tolerant quantum computation with high-rate qLDPC codes that is compatible with the recently demonstrated capabilities of reconfigurable atom arrays. Our approach utilizes the product structure inherent in many qLDPC codes to implement the nonlocal syndrome extraction circuit through atom rearrangement, resulting in an effectively constant overhead. We prove the fault tolerance of these protocols, and our simulations show that the qLDPC-based architecture starts to outperform the surface code with as few as several hundred physical qubits. We further find that quantum algorithms involving thousands of logical qubits can be performed using less than 10^5 physical qubits. Our work suggests that low-overhead quantum computing with qLDPC codes is within reach using current experimental technologies.

Quantum error correction (QEC) is believed to be essential for realizing large-scale fault-tolerant quantum information processing. However, traditional schemes for achieving QEC, such as the paradigmatic surface code, are generally very costly in terms of resource overhead, requiring millions of qubits to solve problems of interest[1–4].

Recently, a new approach based on high-rate quantum low-density parity-check (qLDPC) codes has been proposed as a promising route to reduce the resources required. Unlike planar surface codes[1,2,5] that encode a single logical qubit per block, qLDPC codes can encode many logical qubits per block and achieve a much higher, asymptotically constant, encoding rate[6,7] as well as better distance scaling[8–10]. However, to realize these appealing features, qLDPC codes require long-range connectivity between qubits, making their physical realization challenging[11–13]. Although several proposals have been made for physical implementation of qLDPC codes in superconducting qubit architectures, the required long-range and multi-layer connectivity is considerably beyond both current and medium-term hardware capabilities[14–16].

In bringing qLDPC codes into practical use for full-fledged quantum computation, further challenges arise. A rigorous analysis of the circuit-level fault tolerance of qLDPC codes is lacking, despite some

[1]Pritzker School of Molecular Engineering, The University of Chicago, Chicago, IL, USA. [2]Department of Physics, Harvard University, Cambridge, MA, USA. [3]Institute for Quantum Information and Matter, California Institute of Technology, Pasadena, CA, USA. [4]Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, USA. [5]QuEra Computing Inc., Boston, MA, US. [6]These authors contributed equally: Qian Xu, J. Pablo Bonilla Ataides. ✉e-mail: liang.jiang@uchicago.edu; hyzhou@quera.com

https://doi.org/10.1038/s41567-024-02479-z

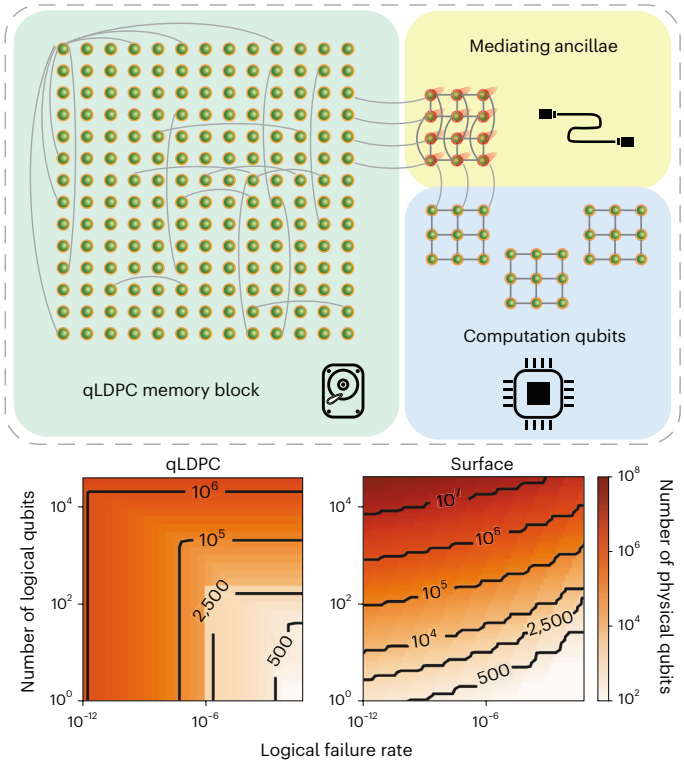


Fig. 1 | Architecture of a qLDPC-based fault-tolerant quantum computer using reconfigurable atom arrays. The computer consists of a qLDPC memory block, a processor with computational logical qubits, and mediating ancillae between the memory and the processor. The lower panel shows a contour plot of the number of physical qubits (including data and ancilla qubits) required by our architecture, at a 10^{-3} physical error rate, given a target number of logical qubits and a target LFR, compared to the surface code. The qLDPC space overhead is given by the minimum of that for the LP codes shown in Fig. 3b with less than 1,428 data qubits and that for HGP codes using an extrapolation of the numerical results in Fig. 3a.

promising numerical evidence[14,15]. Also, it is currently unclear whether finite-size qLDPC codes can outperform surface codes in near- or medium-term devices with $\lesssim$10,000 qubits and realistic physical error rates above 10^{-3}. Since Gottesman's seminal results demonstrating that qLDPC codes can enable fault-tolerant quantum computing with a constant space overhead[17], several practical gate constructions have recently been proposed[18–21]. However, no studies on the circuit-level performance of these protocols have been carried out to date. In particular, it is not clear whether the performance and low overhead of the qLDPC codes can be maintained during computation in a full circuit-level fault-tolerant setting.

In this Article, we propose and analyse a realistic, hardware-efficient, neutral-atom implementation of fault-tolerant quantum computation with high-rate qLDPC codes. We provide concrete experimental and theoretical blueprints and demonstrate their advantage over surface codes starting from as few as several hundred physical qubits. Our proposal is based on reconfigurable atom arrays, a newly developed hardware architecture for quantum computation with long-range, reconfigurable connectivity[22,23]. We show how the product structure of many qLDPC codes[6,8,24] naturally matches the parallelism afforded by physical realizations of reconfigurable atom arrays, which enables their hardware-efficient implementation in a logarithmic number of steps. Through a combination of threshold proofs under the single-shot, circuit-level noise model setting, and detailed circuit-level numerical simulations, we find competitive performance for hypergraph product (HGP)[6] codes and quasi-cyclic lifted product (LP)[8] codes, achieving error thresholds of around 0.6% under a circuit-level, depolarizing-noise model that neglects idling errors. Accounting for idling errors, which have only a minor contribution for the finite-size codes of our interest, we achieve an order of magnitude saving over a surface code with less than 3,000 physical qubits (including ancillae) at a physical error rate of 10^{-3} (Fig. 1 bottom panel and Table 1). We further extend this analysis to logical gate operation, numerically demonstrating that the high thresholds and good subthreshold scaling of high-rate qLDPC codes can be maintained during computation, thus paving the way to low-overhead fault-tolerant quantum computing.

Table 1 | Total number of data and ancilla qubits required to reach target numbers of logical qubits and LFRs using HGP codes and LP codes, compared to using surface codes

Logical qubits	25	80	180	400
Logical failure rate	10^{-3}	10^{-4}	2×10^{-5}	6×10^{-6}
HGP code physical qubits (improvement over surface code)	1,235 (×1)	4,606 (×2.8)	10,760 (×4.0)	19,600 (×6.9)
LP code physical qubits (improvement over surface code)	851 (×1.4)	1,367 (×9.4)	2,670 (×16.2)	

We use (×β) to indicate a β times qubit saving compared to the surface codes by using the corresponding qLDPC codes. The physical error rate is set to 10^{-3}. The estimates for the HGP and LP codes are based on the numerical data in Fig. 3.

Overview of the qLDPC-based quantum computer

An overview of our qLDPC-based approach to fault-tolerant quantum computation is shown in Fig. 1. It consists of a high-rate qLDPC memory block that reliably and efficiently stores the quantum information, a processor with computational logical qubits such as surface or colour codes that perform logical gates, and mediating ancillae that interconnect the memory and processor. This allows us to take advantage of the dense storage capabilities of the qLDPC block while allowing flexible execution of quantum circuits. Adopting the conventional $[\![n, k, d]\!]$ notation for a code with n physical qubits, k logical qubits and distance d, the qLDPC block using the HGP codes described below can provide a dense $[\![\Theta(k), k, d_{\mathrm{mem}}]\!]$ encoding, where the memory distance $d_{\mathrm{mem}} = O(\sqrt{k})$. This results in a constant encoding rate k/n and a logical failure rate (LFR) exponentially decaying with the code distance. Here the LFR is defined as the probability that any of the logical qubits fails per code cycle. O and Θ are standard asymptotic notations denoting less or equal to and equal to (in scaling), respectively. Note that using LP codes with a higher encoding rate and better distance scaling (Methods and Fig. 3) further reduces the space overhead at small sizes. Our processor consists of m computational qubits of code parameters $[\![\Theta(d_{\mathrm{comp}}^2), 1, d_{\mathrm{comp}}]\!]$, where the code distance $d_{\mathrm{comp}} = \Theta(\mathrm{polylog}(kT))$, with T being the depth of the logical circuit to execute, is chosen to produce a sufficiently low error rate. The mediating ancillae, one for each computational qubit (Fig. 4), have code parameters $[\![O(d_{\mathrm{comp}} d_{\mathrm{mem}}), 1, \min(d_{\mathrm{comp}}, d_{\mathrm{mem}})]\!]$. By performing ancilla-assisted lattice surgery, the stored logical information can be teleported between any given pair of memory and computation qubits. Within this architecture, logical gates can be applied in parallel to a subset m of the stored memory qubits in each logical circuit step. Selecting $m = O(k/d_{\mathrm{comp}} d_{\mathrm{mem}})$ ensures that the cumulative ancilla block overhead is $O(k)$, which does not exceed the overhead of the memory block. This choice leads to a constant encoding rate for quantum computation. For HGP codes with $d_{\mathrm{mem}} = \Theta(\sqrt{k})$ and $d_{\mathrm{comp}} = \Theta(\mathrm{polylog}(kT))$, this implies $m = O(\sqrt{k}/\mathrm{polylog}(kT))$, but if a smaller code distance provides sufficient error suppression, the parallelism can be increased while maintaining a constant encoding rate.

https://doi.org/10.1038/s41567-024-02479-z

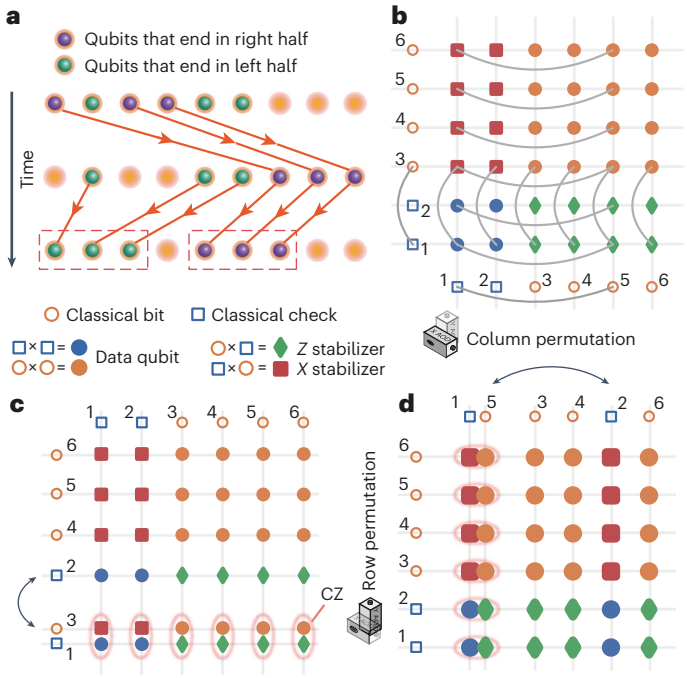


Fig. 2 | Efficient implementation of quantum LDPC codes with atom arrays. **a**, Illustration of the algorithm used to perform an arbitrary 1D log-depth rearrangement. We first move all atoms that need to end in the right half of the system to the right side, then compact each half into adjacent sites, so that there is sufficient workspace for subsequent steps. Atoms are displaced perpendicularly during their movement to avoid collisions with static atoms in the same row (Supplementary Video 1). The same procedure can then be repeated on each half of the system recursively for depth $\log(L)$, where L is the length of the atom array to be rearranged, resulting in the desired ordering. The algorithm uses 50% more static traps than the number of atoms as workspace. **b**, Illustration of the HGP code, obtained as a product of two classical codes. Lines indicate that the parity check at the syndrome node involves the corresponding data node. **c**,**d**, The required connectivity can be implemented with parallel row permutations (**c**) followed by parallel column permutations (**d**). Although we illustrate this for one pair of interacting rows or columns, the same permutations and CZs can be applied on several rows or columns in parallel.

Implementation in neutral-atom arrays

We now describe the hardware-efficient implementation of this qLDPC-based architecture in the atom array platform. Here, qubits are encoded in long-lived spin degrees of freedom of an atom, with seconds-range coherence times[22,25–27], high-fidelity single-qubit (>99.9%) and two-qubit (>99.5%) control, and mid-circuit measurements (>99.8%)[22,25,26,28–30]. By shuttling atoms around in optical tweezers, one can reconfigure the processor connectivity during quantum evolution with minimal decoherence[22,31] and realize parallel two-qubit gate operations with qubits across the whole system. The coherent shuttling approach features a high degree of parallelism, which is inherent to multiplexing with optical tools. A particularly powerful tool is the so-called acousto-optic deflector (AOD), which can simultaneously control the position of rectangular grids composed of thousands of atoms simply by using two control waveforms for the X and Y coordinates[22]. These AOD tools are a key enabling technology for qubit transport in atom arrays and support an inherent 'product structure' that, as we will now show, is suited well to the implementation of HGP and LP codes.

We first describe an algorithm to implement good classical LDPC codes using efficient one-dimensional (1D) atom rearrangements without atom collisions. Recall that a classical (LDPC) code is associated with a parity-check matrix H, whose rows (columns) are associated with classical checks (bits). The ith check is connected to the jth bit if $H_{i,j} = 1$. We first lay out the classical LDPC code on a line and group syndrome extraction into layers of parallel entangling operations, resulting in an ordering of checks and bits such that the connected checks and bits in a given layer are neighbouring. To achieve this ordering, we employ a divide-and-conquer rearrangement strategy that requires a number of steps only logarithmic in the total number of checks and bits, as described in detail in Fig. 2a, Extended Data Fig. 2, Methods and Supplementary Video 1. This generalizes previous proposals for scrambling circuits[32] to arbitrary rearrangements. After sorting the atoms, a global laser pulse is applied to entangle neighbouring checks and bits in parallel, before proceeding to the next layer of atom rearrangement, thus allowing the implementation of syndrome extraction in classical LDPC codes.

We next demonstrate how the product structure of one of the prototypical qLDPC codes (Fig. 2b and Extended Data Fig. 1a), HGP codes, naturally matches the product structure of crossed AOD optical hardware, enabling its hardware-efficient implementation. We start from a pair of classical LDPC codes illustrated in the horizontal and vertical directions, with checks and bits denoted as blue squares and orange circles, respectively. To construct the resulting HGP code, we place a data qubit at each intersection of a horizontal check and a vertical check or a horizontal bit and a vertical bit, within a two-dimensional grid[6] (Fig. 2b). X stabilizer syndrome qubits and Z stabilizer syndrome qubits are then placed at the intersection of a horizontal check and a vertical bit or a vertical check and a horizontal bit, respectively. Importantly, the qubit connectivities are directly inherited from the underlying classical code in the horizontal and vertical directions, and thus, the same entangling gates are applied across every row or column, matching well with the product structure of crossed AODs. Thus, by performing parallel row reordering in the vertical direction based on the vertical LDPC code interleaved with entangling gates between data and stabilizer qubits (Fig. 2c) and then repeating the same in the horizontal direction (Fig. 2d), we can implement the syndrome extraction for HGP codes. The concrete syndrome extraction circuits are presented in Algorithms 1–3 in Methods.

We now estimate the scaling and quantitative experimental timescales of our rearrangement algorithm and demonstrate that the proposed hardware implementation is indeed achievable with existing experimental parameters. In Methods, we show that the total rearrangement time scales as $O(\sqrt{L}) = O(\sqrt[4]{n})$, where L is the length of the two-dimensional atom array and is like the scaling for a constant acceleration trajectory. We estimate that for a moderately sized HGP code with 10,000 qubits, each rearrangement layer between gates requires 3 ms, a small fraction of the coherence time >10 s that has been demonstrated for neutral-atom arrays[25–27]. These timescales can be substantially improved through innovations in optical technologies and compilation. Ancilla measurements can be pipelined, thereby not causing any increase in the cycle time. For very large codes, where idling errors are no longer negligible, concatenation with another code can also be employed to extend the effective coherence time[33]. Note that HGP codes based on expanding classical LDPC codes (also called quantum expander codes) have the single-shot property, and therefore, only a single round of syndrome extraction is required to be fault-tolerant[34–36]. Although we have focused on the implementation of HGP codes, other families of qLDPC codes, such as LP codes (Extended Data Fig. 1b), can also be implemented by adapting similar ideas. See Methods for details.

qLDPC memory

We now analyse the fault-tolerant implementation of HGP and LP codes as a robust quantum memory. We prove in Supplementary Information the existence of a circuit-level single-shot threshold for qLDPC codes with the linear confinement property[35], under a single-ancilla syndrome extraction circuit and a depolarizing-noise model where error rates do not scale with instance size. The linear confinement property, which requires that for sufficiently small Pauli errors the weight of

https://doi.org/10.1038/s41567-024-02479-z

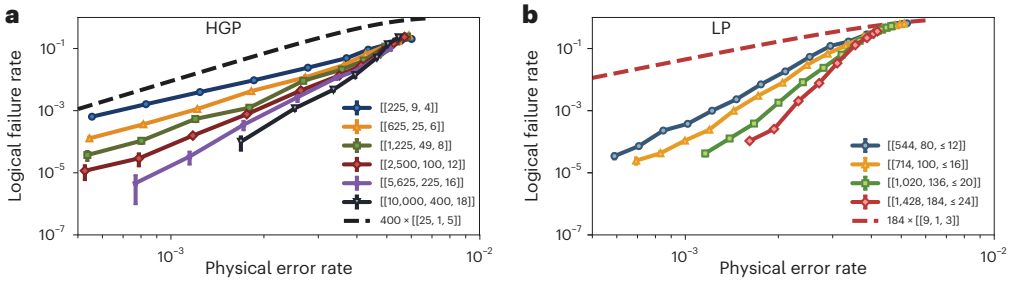


Fig. 3 | qLDPC memory performance. a,**b**, LFRs as a function of physical error rate for the qLDPC memory using HGP codes (**a**) and LP codes (**b**), for a depolarizing error model that includes idling errors that increase with the code size. For the HGP codes, we take the HGP of classical codes associated with random (3,4)-regular bipartite graphs that have good expanding properties, which have an encoding rate lower-bounded by 1/25. For the LP codes, we choose 3 by 5 base matrices over a quotient polynomial ring and obtain a family of codes with sizes up to 1,428 by increasing the lift size and optimizing the matrix entries. These LP codes have an encoding rate lower-bounded by 2/17 and maintain a higher encoding rate as well as better distances than HGP codes of the same size. See Methods for details of the code construction. We use a product coloration circuit for syndrome extraction (Algorithm 2) and a space–time circuit-level decoder based on BP+OSD that decodes over every three code cycles, regardless of the code size. The LFRs were calculated using $\mathrm{LFR} = 1-(1-p_\mathrm{L})^{1/m_\mathrm{c}}$, where p_L is the total logical failure probability over m_c code cycles. We choose $m_\mathrm{c} = 42$ for physical error rates below 4×10^{-3} and $m_\mathrm{c} = 12$ for physical error rates above 4×10^{-3}. p_L was obtained from Monte Carlo simulations with a standard deviation $\sqrt{p_\mathrm{L}(1-p_\mathrm{L})/M}$, where M denotes the number of samples. We also compare the largest HGP or LP code to surface codes of similar sizes and encoding rates (dashed lines), as discussed in 'Main'. Data are presented as mean values plus or minus standard deviations.

the syndrome increases linearly with the (reduced) weight of the errors, holds for various qLDPC codes decodable by the small-set-flip-type decoders, including HGP codes with sufficiently expanding classical codes.

We next supplement this theoretical understanding with numerical simulations of HGP and LP codes at practically relevant instance sizes[37], and we find competitive thresholds and LFRs for both codes. The details of the code constructions are shown in Fig. 2 and Methods. We use the product coloration circuit (Algorithm 2) for syndrome extraction, a variation of the coloration circuit[14] that is more compatible with the product structure of the codes and hardware. The circuit uses a single ancilla for each stabilizer generator and has an entangling-gate depth of 16 and 20 for the HGP and LP codes we consider. Note that the entangling depth can be further reduced by using a pipelined version of the product coloration circuit, as presented in Extended Data Fig. 3 and Methods.

We construct a space–time circuit-level decoder based on the belief propagation and ordered statistics decoding (BP+OSD) algorithm[7,38–40]. Specifically, for a QEC circuit with several cycles, we construct a bipartite decoding graph[39,41,42] over a certain number of cycles, where the check nodes and variable nodes are associated with parities of stabilizer measurement outcomes and circuit faults, respectively. We apply BP decoding on this decoding graph to infer the circuit fault locations in all noisy code cycles and apply the BP+OSD decoder in the final round to project back into the code space. For all memory simulations, we use space–time decoding graphs over three cycles, irrespective of the code size. Note that we observe improved logical error rates by increasing the number of cycles for decoding, which indicates that there is a trade-off between the accuracy and the speed of our decoder. We also remark that the BP decoder has similar performance as the BP+OSD decoder for decoding the noisy cycles while being much faster. Crucially, compared to earlier phenomenological decoders that used a simpler decoding graph involving only independent data and measurement errors as bits and decoded over only one code cycle[15,43], our space–time circuit-level decoder takes the full circuit details into account and can perform joint decoding on several QEC cycles, thus improving the threshold (Supplementary Information). Moreover, the space–time decoding is also crucial for our simulations of logical operations in the next section, where repetitions of syndrome measurements are required for fault tolerance. See Supplementary Information for details of the decoder.

Supplementary Information shows that the HGP and LP codes have a threshold of 0.63% and 0.62%, respectively, under a depolarizing error model without idling errors. In the subthreshold regime, assuming the absence of a decoding error floor, the LFRs of the two codes are well approximated by

$$\begin{aligned}\mathrm{LFR(HGP)} &= 0.07(p_\mathrm{g}/0.006)^{0.47n^{0.27}},\\ \mathrm{LFR(LP)} &= 2.3(p_\mathrm{g}/0.0066)^{0.11n^{0.60}},\end{aligned} \tag{1}$$

indicating that finite-sized LP codes have better subthreshold scaling than HGP codes. When also considering idling errors $p_i(n)$ associated with the atom-rearrangement time overhead, which grow as $O(n^{1/4})$ for HGP codes and $O(n^{1/2})$ for LP codes (see Methods for details of the idling error model and the expression for $p_i(n)$ in equation (8)), there is no asymptotic code threshold[15]. However, we numerically observe that the effect of adding the idling errors can be approximated by rescaling the gate error $p_\mathrm{g} \to p_\mathrm{g} + 3p_i(n)$ using the product coloration circuit (Supplementary Information). The idling errors have a negligible contribution when $3p_i(n) \ll p_\mathrm{g}$, which is the case for current experimental parameters and practically relevant code sizes (Extended Data Fig. 4 and Methods). Therefore, although there is no asymptotic threshold, constant overhead and fault tolerance can still be achieved at physically relevant sizes by utilizing the quasi-nonlocal connectivity in atom arrays. Figure 3a,b shows the simulated LFR versus the bare two-qubit gate error rate p_g (which we refer to as the physical error rate for simplicity) for the HGP and LP codes, with the idling errors included and rescaled together with p_g. We find that good LFRs and subthreshold scaling are maintained for both codes in the presence of idling errors. The smallest instances of these codes that we simulate, involving hundreds of physical qubits, are readily within reach with experimentally demonstrated system sizes[44] and control capabilities[22,23], and larger instances can be realized with improvements in trap power.

We now use equation (1) to estimate the total number of data and ancilla qubits N needed to reach a target number of logical qubits k and a target LFR, demonstrating substantial advantages of HGP and LP codes over the surface code. We rescale the gate error p_g to approximate the presence of idling errors and compare the results with the surface-code subthreshold scaling formula $\mathrm{LFR(surface)} = 1-(1-P_0)^k$, where $P_0 = 0.03(p_\mathrm{g}/0.011)^{\lfloor|\sqrt{n/k}\rfloor/2|}$, from refs. 1,45. Table 1 presents such estimates for finite-sized HGP and LP codes that we directly simulated (Fig. 3) at a realistic physical error rate of 10^{-3}. Both HGP and LP codes outperform surface codes with as

https://doi.org/10.1038/s41567-024-02479-z

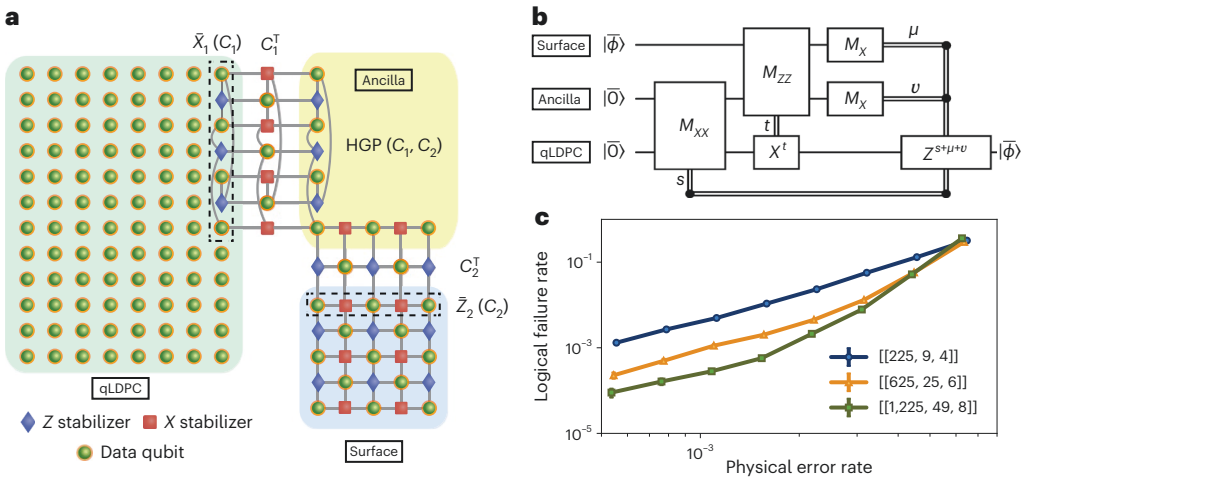


Fig. 4 | Fault-tolerant teleportation from surface to qLDPC code. a, Illustration of the teleportation scheme. We identify the logical Z operator of the surface code $\bar{Z}_2$ and the logical X operator of one of the qLDPC code's logical qubits $\bar{X}_1$. We associate these two logical operators with classical codes C_1 and C_2 by mapping the qubits supporting the logical operators to bits and the corresponding incident stabilizer generators to classical checks. We then construct an ancilla patch as the HGP of C_1 and C_2. Direct lattice surgery between this ancilla patch and each of the surface and HGP codes is conducted by matching similar boundaries associated with the chosen logical operators. In between similar boundaries, an extra array of ancillary qubits and checks associated with the transpose of the classical code is inserted to mediate the surgery. The product of the stabilizers associated with the checks of the transposed code gives the required joint logical measurement. We elaborate on the lattice-surgery procedure in Methods. **b**, Measurement-based teleportation circuit[47]. Logical state $|\bar{\phi}\rangle$ is teleported from the surface code to one of the qLDPC's logical qubits. The joint Clifford measurements are conducted through lattice surgery as illustrated in **a**. **c**, Simulated LFRs (per code cycle) of the teleportation. Noise is added during the merge and split steps of the XX lattice surgery. We decode with the same space–time circuit-level BP+OSD decoder used in the memory simulations. The corresponding surface codes paired with the three HGP codes have distances 3, 5 and 7. We record a logical failure if there is an error in any of the logical qubits of the qLDPC code after the teleportation scheme is complete. Denoting the total logical failure probability as p_L, we calculate the LFR (per code cycle) as LFR $= 1-(1-p_L)^{2d}$, where there are $2d$ cycles during the noisy XX lattice surgery, and d denotes the minimal code distance. The plotted physical error rates are rescaled to account for idling errors, as explained in Methods. Data are presented as mean values plus or minus standard deviations.

few as 25 logical qubits. At a moderate scale of less than 200 logical qubits, LP codes with less than 3,000 physical qubits already achieve a qubit saving of over an order of magnitude. In the lower panel of Fig. 1, we estimate the space overhead of the HGP codes at a larger scale by extrapolation. We find that HGP codes can also achieve a qubit saving of over an order of magnitude at a scale of 1,000 logical qubits and 10^5 physical qubits.

Logical operations

We now present a scheme inspired by ref. 18 for performing fault-tolerant logical operations and perform the first numerical simulation of logical gate performance on qLDPC codes using the space–time decoder developed above. We find that the threshold and logical performance remain almost unchanged when performing gates, indicating that the high threshold and low overhead can be maintained for fault-tolerant quantum computation.

Our scheme is illustrated in Fig. 4a. We teleport the logical information between the qLDPC memory and ancillary topological codes using a measurement-based circuit (Fig. 4b), where the prescribed logical measurements are implemented using lattice surgery[18,46–48]. Universal logical operations can then be performed in the topological codes using standard techniques. Our approach is applicable to any CSS qLDPC code, as proven in Supplementary Information. The choice of qLDPC code, whether an HGP or an LP code, does not impact the feasibility of our ancilla encoding for logical operations. Since each topological code patch and teleportation ancilla patch is much smaller than the qLDPC patch, as long as the number of such patches used is $O(\sqrt{k}/\mathrm{poly}\log(kT))$, the space overhead from ancilla patches will be sub-leading. Note that the scheme in ref. 18 can also be used for teleportation between a qLDPC block and topological codes. In comparison, our scheme reduces the ancilla patch size by half and, thus, has a smaller space overhead.

As an example, we consider the teleportation from a surface code patch to a HGP patch. As illustrated in Fig. 4a, this is mediated by an additional ancilla logical qubit that is formed by a HGP of two classical codes associated with the logical operators of the two code patches. This enables joint logical measurements by merging code blocks. We perform $\min\{d_{\mathrm{comp}}, d_{\mathrm{mem}}\}$ rounds of syndrome extraction to ensure tolerance against measurement errors. We describe the scheme in more detail in Methods and prove the fault tolerance of the scheme under data errors in Supplementary Information.

To validate this method, we perform circuit-level simulations of the above teleportation, enabled by the space–time decoder described in the previous section. In our simulations, we use the teleportation circuit depicted in Fig. 4b with an initial state of $|\psi\rangle = |0\rangle$. We focus on errors during the merge and split operations in the XX measurement to evaluate the performance of lattice-surgery building blocks. As shown in Fig. 4c, we observe similar LFRs and threshold crossings for the teleportation as for the memory. Note that each logical qubit will fail at a rate much lower than the LFR of the entire block shown in Fig. 4c, and the system is far below break-even at 10^{-3} physical error rate, even for the smallest code of size 225. This demonstrates that the high-threshold and low-resource overhead of the qLDPC code can be maintained at the computation level.

Discussion and outlook

By demonstrating large space-overhead savings in practical regimes and the good performance of logical gate operations and by providing a blueprint for their implementation with existing hardware capabilities, our work brings the use of high-rate qLDPC codes for fault-tolerant quantum computation into the practical regime.

Although our scheme shows a notable reduction in space overhead, it still carries a substantial time overhead. This is because fault tolerance during gate operation requires $\Theta(d)$ QEC cycles, and the low

encoding rate of the ancilla and computational code patches limits the logical parallelism when maintaining the low space overhead. We expect certain compilations of quantum algorithms that have limited parallelism to be natural candidates for our architecture[4]. However, it would be interesting to carry out an end-to-end algorithmic compilation with qLDPC codes to evaluate the full space–time cost and understand which algorithms and compilations are most suited to our architecture. Another exciting avenue of research is to improve upon the QEC constructions used here, including alternative qLDPC code constructions with better properties[49–52], single-shot logical gate constructions[20,21], as well as the use of other types of computational logical qubits that may support transversal non-Clifford gates[53] or have a lower overhead. Moreover, just as topological codes have interpretations as topological phases of matter, it will be interesting to explore the connection between qLDPC codes and highly entangled states of matter[54]. The techniques described here may be useful for exploring novel exotic states of matter.

Online content

Any methods, additional references, Nature Portfolio reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41567-024-02479-z.

References

1. Fowler, A. G., Mariantoni, M., Martinis, J. M. & Cleland, A. N. Surface codes: towards practical large-scale quantum computation. *Phys. Rev. A* **86**, 032324 (2012).
2. Litinski, D. A game of surface codes: large-scale quantum computing with lattice surgery. *Quantum* **3**, 128 (2019).
3. Beverland, M. E. et al. Assessing requirements to scale to practical quantum advantage. Preprint at https://arxiv.org/abs/2211.07629 (2022).
4. Gidney, C. & Ekerå, M. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum* **5**, 433 (2019).
5. Kitaev, A. Y. Fault-tolerant quantum computation by anyons. *Ann. Phys.* **303**, 2–30 (2003).
6. Tillich, J. P. & Zemor, G. Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength. *IEEE Trans. Inf. Theory* **60**, 1193–1202 (2014).
7. Panteleev, P. & Kalachev, G. Degenerate quantum LDPC codes with good finite length performance. *Quantum* **5**, 585 (2019).
8. Panteleev, P. & Kalachev, G. Quantum LDPC codes with almost linear minimum distance. *IEEE Trans. Inf. Theory* **68**, 213–229 (2022).
9. Breuckmann, N. P. & Eberhardt, J. N. Balanced product quantum codes. *IEEE Trans. Inf. Theory* **67**, 6653–6674 (2020).
10. Panteleev, P. & Kalachev, G. Asymptotically good quantum and locally testable classical LDPC codes. In *Proc. Annual ACM Symposium on Theory of Computing* 375–388 (ACM, 2022).
11. Bravyi, S., Poulin, D. & Terhal, B. Tradeoffs for reliable quantum information storage in 2D systems. *Phys. Rev. Lett.* **104**, 050503 (2010).
12. Baspin, N. & Krishna, A. Connectivity constrains quantum codes. *Quantum* **6**, 711 (2021).
13. Baspin, N. & Krishna, A. Quantifying nonlocality: how outperforming local quantum codes is expensive. *Phys. Rev. Lett.* **129**, 050505 (2022).
14. Tremblay, M. A., Delfosse, N. & Beverland, M. E. Constant-overhead quantum error correction with thin planar connectivity. *Phys. Rev. Lett.* **129**, 050504 (2022).
15. Delfosse, N., Beverland, M. E. & Tremblay, M. A. Bounds on stabilizer measurement circuits and obstructions to local implementations of quantum LDPC codes. Preprint at https://arxiv.org/abs/2109.14599v1 (2021).
16. Strikis, A. & Berent, L. Quantum low-density parity-check codes for modular architectures. *PRX Quantum* **4**, 020321 (2023).
17. Gottesman, D. Fault-tolerant quantum computation with constant overhead. *Quantum Inf. Comput.* **14**, 1338–1372 (2013).
18. Cohen, L. Z., Kim, I. H., Bartlett, S. D. & Brown, B. J. Low-overhead fault-tolerant quantum computing using long-range connectivity. *Sci. Adv.* **8**, eabn1717 (2022).
19. Krishna, A. & Poulin, D. Fault-tolerant gates on hypergraph product codes. *Phys. Rev. X* **11**, 011023 (2021).
20. Breuckmann, N. P. & Burton, S. Fold-transversal Clifford gates for quantum codes. Preprint at https://arxiv.org/abs/2202.06647v2 (2022).
21. Quintavalle, A. O., Webster, P. & Vasmer, M. Partitioning qubits in hypergraph product codes to implement logical gates. Preprint at https://arxiv.org/abs/2204.10812v2 (2022).
22. Bluvstein, D. et al. A quantum processor based on coherent transport of entangled atom arrays. *Nature* **604**, 451–456 (2022).
23. Bluvstein, D. et al. Logical quantum processor based on reconfigurable atom arrays. *Nature* **626**, 58–65 (2024).
24. Breuckmann, N. P. & Eberhardt, J. N. Quantum low-density parity-check codes. *PRX Quantum* **2**, 040101 (2021).
25. Jenkins, A., Lis, J. W., Senoo, A., McGrew, W. F. & Kaufman, A. M. Ytterbium nuclear-spin qubits in an optical tweezer array. *Phys. Rev. X* **12**, 021027 (2022).
26. Ma, S. et al. Universal gate operations on nuclear spin qubits in an optical tweezer array of ^{171}Yb atoms. *Phys. Rev. X* **12**, 021028 (2022).
27. Barnes, K. et al. Assembly and coherent control of a register of nuclear spin qubits. *Nat. Commun.* **13**, 2779 (2022).
28. Evered, S. J. et al. High-fidelity parallel entangling gates on a neutral atom quantum computer. *Nature* **622**, 268–272 (2023).
29. Ma, S. et al. High-fidelity gates with mid-circuit erasure conversion in a metastable neutral atom qubit. Preprint at https://arxiv.org/abs/2305.05493v1 (2023).
30. Scholl, P. et al. Erasure-cooling, control, and hyper-entanglement of motion in optical tweezers. Preprint at https://arxiv.org/abs/2311.15580v1 (2023).
31. Beugnon, J. et al. Two-dimensional transport and transfer of a single atomic qubit in optical tweezers. *Nat. Phys.* **3**, 696–699 (2007).
32. Hashizume, T., Bentsen, G. S., Weber, S. & Daley, A. J. Deterministic fast scrambling with neutral atom arrays. *Phys. Rev. Lett.* **126**, 200603 (2021).
33. Pattison, C. A., Krishna, A. & Preskill, J. Hierarchical memories: simulating quantum LDPC codes with local gates. Preprint at https://arxiv.org/abs/2303.04798 (2023).
34. Leverrier, A., Tillich, J. P. & Zemor, G. Quantum expander codes. In *Proc. Annual IEEE Symposium on Foundations of Computer Science* 810–824 (IEEE, 2015).
35. Quintavalle, A. O., Vasmer, M., Roffe, J. & Campbell, E. T. Single-shot error correction of three-dimensional homological product codes. *PRX Quantum* **2**, 020340 (2020).
36. Bombin, H. Single-shot fault-tolerant quantum error correction. *Phys. Rev. X* **5**, 031043 (2015).
37. Gidney, C. Stim: a fast stabilizer circuit simulator. *Quantum* **5**, 497 (2021).
38. Roffe, J., White, D. R., Burton, S. & Campbell, E. Decoding across the quantum low-density parity-check code landscape. *Phys. Rev. Res.* **2**, 043423 (2020).
39. Higgott, O. & Breuckmann, N. P. Improved single-shot decoding of higher-dimensional hypergraph-product codes. *PRX Quantum* **4**, 020332 (2023).
40. Kuo, K. Y. & Lai, C. Y. Exploiting degeneracy in belief propagation decoding of quantum codes. *npj Quantum Inf.* **8**, 111 (2022).

41. Higgott, O., Bohdanowicz, T. C., Kubica, A., Flammia, S. T. & Campbell, E. T. Improved decoding of circuit noise and fragile boundaries of tailored surface codes. *Phys. Rev. X* **13**, 031007 (2023).
42. Delfosse, N. & Paetznick, A. Spacetime codes of Clifford circuits. Preprint at https://arxiv.org/abs/2304.05943v2 (2023).
43. Grospellier, A., Grouès, L., Krishna, A. & Leverrier, A. Combining hard and soft decoders for hypergraph product codes. *Quantum* **5**, 432 (2021).
44. Ebadi, S. et al. Quantum optimization of maximum independent set using Rydberg atom arrays. *Science* **376**, 1209–1215 (2022).
45. Wang, D. S., Fowler, A. G. & Hollenberg, L. C. L. Surface code quantum computing with error rates over 1. *Phys. Rev. A* **83**, 020302 (2011).
46. Horsman, C., Fowler, A. G., Devitt, S. & van Meter, R. Surface code quantum computing by lattice surgery. *New J. Phys.* **14**, 123011 (2012).
47. Breuckmann, N. P., Vuillot, C., Campbell, E., Krishna, A. & Terhal, B. M. Hyperbolic and semi-hyperbolic surface codes for quantum storage. *Quantum Sci. Technol.* **2**, 035007 (2017).
48. Poulsen Nautrup, H., Friis, N. & Briegel, H. J. Fault-tolerant interface between quantum memories and quantum processors. *Nat. Commun.* **8**, 1321 (2017).
49. Bravyi, S. et al. High-threshold and low-overhead fault-tolerant quantum memory. *Nature* **627**, 778–782 (2024).
50. Hong, Y., Marinelli, M., Kaufman, A. M. & Lucas, A. Long-range-enhanced surface codes. Preprint at https://arxiv.org/abs/2309.11719v1 (2023).
51. Fahimniya, A. et al. Fault-tolerant hyperbolic Floquet quantum error correcting codes. Preprint at https://arxiv.org/abs/2309.10033v2 (2023).
52. Higgott, O. & Breuckmann, N. P. Constructions and performance of hyperbolic and semi-hyperbolic Floquet codes. Preprint at https://arxiv.org/abs/2308.03750v1 (2023).
53. Bombín, H. Gauge color codes: optimal transversal gates and gauge fixing in topological stabilizer codes. *New J. Phys.* **17** 083002 (2015).
54. Anshu, A., Breuckmann, N. P. & Nirkhe, C. NLTS Hamiltonians from good quantum codes. Preprint at https://arxiv.org/abs/2206.13228v2 (2022).

Methods

Code construction

We primarily focus on two families of qLDPC codes, outlined here with additional information available in Supplementary Information. We leave the extension of these results to other families, such as asymptotically good codes[8–10,55–58], to future work.

The first family of codes are HGP codes[6], which are formed from the product of two classical LDPC codes. Algebraically, if we denote the parity-check matrix (where rows describe bits that should sum to an even number in the absence of errors) of the two underlying classical codes as $H_1 \in \mathbb{F}_2^{r_1 \times n_1}$ and $H_2 \in \mathbb{F}_2^{r_2 \times n_2}$, then the X and Z stabilizer check matrices for the HGP code can be written as

$$H_x = (H_1^T \otimes I_{r_2} \;\; I_{n_1} \otimes H_2), \tag{2}$$

$$H_z = (I_{r_1} \otimes H_2^T \;\; H_1 \otimes I_{n_2}). \tag{3}$$

For classical $[n_i, k_i, d_i]$ linear codes defined by $r_i = n_i - k_i$ linearly independent checks ($i = 1, 2$), the resulting quantum code has parameters $[\![n_1 n_2 + r_1 r_2, k_1 k_2, \min\{d_1, d_2\}]\!]$. By choosing classical codes with good vertex expansion, where $k_i = \Theta(n_i)$ and $d_i = \Theta(n_i)$, the resulting quantum code (known as a quantum expander code) encodes a linear number of logical qubits $k = \Theta(n)$ and has distance $d = \Theta(\sqrt{n})$ (ref. 34). Such classical expander codes can be obtained asymptotically, for example, from random biregular Tanner graphs, and will have sufficient vertex expansion with high probability[59].

In this work, we follow the procedure of ref. 43 and construct HGP codes by taking the HGP of classical LDPC codes defined by (3,4)-regular Tanner graphs, that is, bipartite graphs with degree-3 bit nodes and degree-4 check nodes. By increasing the size of the graph, we obtain a family of HGP codes with a constant encoding rate $k/n \geq 0.04$. For each code size, we pick the classical code with the largest distance, Tanner graph girth at least 6 (length of the shortest cycle in the Tanner graph, obtained through rejection sampling without performing edge swaps) and the largest spectral gap (the gap between the largest two singular values of the classical check matrices) from randomly generated instances. It is known that the HGP of vertex-expanding classical codes gives HGP codes that satisfy the confinement property and support single-shot QEC (refs. 34,35,60).

The second family of codes we consider are quasi-cyclic LP codes[7,9,61], which can be viewed as a HGP code followed by a symmetry reduction to reduce the number of qubits required[24]. Algebraically, a quasi-cyclic LP code is obtained from two base protographs (analogues of the classical codes in the HGP construction) associated with two base matrices B_1 and B_2 over the quotient polynomial ring $\mathbb{R}[x]/(x^l - 1)$ (ref. 8). Suppose the two base matrices are of size $m_{B_1} \times n_{B_1}$ and $m_{B_2} \times n_{B_2}$, respectively. We obtain two matrices (over the same polynomial ring) B_x and B_z by taking the HGP:

$$\begin{aligned} B_x &= \left(B_1^{\mathsf{T}} \otimes I_{m_{B_2}} \;\; I_{n_{B_1}} \otimes B_2\right), \\ B_z &= \left(I_{m_{B_1}} \otimes B_2^{\mathsf{T}} \;\; B_1 \otimes I_{n_{B_2}}\right). \end{aligned} \tag{4}$$

The X (Z) check matrix H_x (H_z) is then obtained by replacing each entry of B_x (B_z) with its matrix representation as l by l circulant matrices, a process known as a lift. Specifically, we replace x with a l by l square matrix P, where $P_{ij} = \delta_{i,j+1}$, and replace each polynomial in x with the same polynomial in P. The code size is $N = l(n_{B_1} n_{B_2} + m_{B_1} m_{B_2})$ and the numbers of X and Z checks are $M_x = l n_{B_1} m_{B_2}$ and $M_z = l m_{B_1} n_{B_2}$, respectively. The encoding rate is lower-bounded by

$$\begin{aligned} (N - M_x - M_z)/N = &\left(n_{B_1} n_{B_2} + m_{B_1} m_{B_2}\right. \\ &\left. - m_{B_1} n_{B_2} - n_{B_1} m_{B_2}\right)/(n_{B_1} n_{B_2} + m_{B_1} m_{B_2}). \end{aligned}$$

We can also describe the above construction using graphs. As an example, Extended Data Fig. 1b shows a LP code using a 3 by 5 protograph associated with a base matrix $B \in \{\mathbb{R}[x]/(x^2 - 1)\}^{3\times 5}$. An important feature of the LP codes is that they still have some remaining product structure, even after the lift. As shown in Extended Data Fig. 1b, when flattening the inner nodes vertically (horizontally), the vertical (horizontal) connectivity between the qubits and the checks for each column (row) is the same as the left (top) lifted classical code.

For the LP codes constructed in this work, we choose a base matrix of dimension 3 by 5, where all entries are monomials, and obtain a family of codes with sizes up to 1,428 by increasing the lift size l from 16 to 42. The classical parity checks are optimized by choosing the base matrix entries over the quotient polynomial ring to obtain the best classical distance for the particular lift size l. The choice of the base matrix entries is also such that the girth is at least 8 and the distance of the lifted qLDPC codes matches the designed classical distances with a high probability. Here, we explicitly provide the classical base matrices used to construct the four LP codes used in this work. Denoting B_d^l as a base matrix with a lift size l and a classical code distance d after the lift, the base matrices are

$$B_{12}^{16} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & x^2 & x^4 & x^7 & x^{11} \\ 1 & x^3 & x^{10} & x^{14} & x^{15} \end{bmatrix}, \quad B_{16}^{21} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & x^4 & x^5 & x^7 & x^{17} \\ 1 & x^{14} & x^{18} & x^{12} & x^{11} \end{bmatrix}, \tag{5}$$

$$B_{20}^{30} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & x^2 & x^{14} & x^{24} & x^{25} \\ 1 & x^{16} & x^{11} & x^{14} & x^{13} \end{bmatrix}, \quad B_{24}^{42} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & x^6 & x^7 & x^9 & x^{30} \\ 1 & x^{40} & x^{15} & x^{31} & x^{35} \end{bmatrix}. \tag{6}$$

These codes have an encoding rate lower-bounded by 2/17 by counting the number of qubits minus the number of checks. For all the resource estimates involving these LP codes, we use $k \approx 0.38 n^{0.85}$, as it fits well for the above four codes.

Atom-rearrangement algorithm

The reconfigurable atom array platform features efficient, parallel control and allows the rearrangement of large numbers of qubits, thus enabling the implementation of quantum processors with long-range connectivity. As discussed in 'Main', optical tools such as crossed AODs can generate a rectangular grid of optical tweezers that can be reconfigured on the fly[22,31,62], which allows the control of large code blocks consisting of thousands of physical qubits with only a handful of classical controls.

However, the use of AODs for a dynamic rearrangement comes with two key constraints. First, as the X and Y direction optical spots are controlled by separate AODs, the same operation needs to be applied across several rows or columns. Second, different rows of atoms cannot cross each other due to beating between RF tones and atom collisions, although they can be temporarily transferred and stored in static traps, such as those based on spatial light modulators. Thus, the implementation of qLDPC codes with improved code parameters (number of encoded qubits and code distance), which often relies on pseudorandom expander graphs with complex connectivity graphs, requires the development of efficient atom-rearrangement algorithms.

We provide a sketch of our atom-rearrangement algorithm in 'Main' and a more detailed description in Algorithms 1–3. We also illustrate the algorithm in Supplementary Video 1.

The first component, arbitrary 1D atom rearrangements with a number of steps that scales logarithmically, is described in detail in Algorithm 1, illustrated in Fig. 2 and explicitly worked out for a small example in Extended Data Fig. 2. As successive layers each reduce the system size by half, the total number of layers required to achieve the desired rearrangement is $\lceil \log_2 L \rceil$. Thus, arbitrary rearrangements in very large systems can be achieved in a small number of layers.

Algorithm 1. Arbitrary 1D atom rearrangement in a logarithmic number of steps

Input : Final ordering $O = [o_i]$ $(i = 1...N)$ of all N atoms, where o_i is the final position of the atom that was initially at position i.
Input : Initial positions $A = [a_i]$ $(i = 1...N)$ of all N atoms, with the positions ordered as $a_1 < a_2 < ... < a_N$.
Input : Positions $P = [p_j]$ $(j = 1...M)$ of all M possible qubit locations, where $M \geq 3N/2$.
Output : Positions $C = [c_{s,i}]$ for the ith atom in the sth rearrangement step, for all N atoms and all rearrangement steps.

Function `Rearrange` (O, A, P) **:**
  if $N = 1$ **then**
    $c_{1,1} \leftarrow p_1$
    return $[c_{s,i}]$
  $s \leftarrow 1$ `// Layer counter`
  if $a_N > p_N$ **then**
    `// Compactify atoms to the left to make space for subsequent moves`
    for $i \leftarrow 1$ **to** N **do**
      $c_{s,i} \leftarrow p_i$
    $s \leftarrow s + 1$
  `// Determine whether each atom ends in the left or right half`
  $L, R \leftarrow []$
  `// Workspace separator for recursion`
  $X = \lfloor 3N/2 \rfloor$
  for $i \leftarrow 1$ **to** N **do**
    if $O_i \leq \lfloor N/2 \rfloor$ **then**
      L.append(i)
      $c_{s,i} \leftarrow p_{N+len(L)}$ `// Move to right`
      $c_{s+1,i} \leftarrow X + len(L)$ `// Compactify`
    else
      R.append(i)
      $c_{s,i} \leftarrow a_i$ `// Stay in place`
      $c_{s+1,i} \leftarrow len(R)$ `// Compactify`
  $s \leftarrow s + 2$
  `// Recursive call on each half`
  $C_l \leftarrow$ `Rearrange` $(O[L], C[s-1, L], P[1..X])$
  $C_r \leftarrow$ `Rearrange` $(O[R], C[s-1, R], P[X+1..M])$
  $C[s.., L] \leftarrow C_l$
  $C[s.., R] \leftarrow C_r$
  return C
`// Main function`
`Rearrange` (O, A, P)

The second component is the observation that the product structure of crossed AODs matches well with the product structure present in many qLDPC codes. In addition to the discussion in 'Main', we provide the details of the syndrome extraction circuit for HGP codes based on this observation in Algorithm 2, which we refer to as the 'product coloration circuit', as it makes use of coloration circuits for each of the component classical codes[14]. Note that the use of our product coloration circuit, as opposed to the coloration or cardinal circuits in ref. 14, is necessary to fully exploit the parallel rearrangement capabilities across rows and columns. Here, the native entangling-gate set of current atom array systems are diagonal[28,63], so we use CZ gates and appropriate Hadamard rotations to perform syndrome extraction. To compare our results against the literature, we use CNOT gates as the entangling gates in our simulations. This can be physically justified if the CZ gates are much noisier than the Hadamard gates.

The product coloration circuit separately extracts the X and Z syndromes, each requiring both a horizontal and vertical step. Thus, if the coloration of each of the classical codes involves Δ_c colours (for the codes constructed from (3,4)-biregular graphs that we considered, $\Delta_c = 4$; ref. 15), the product coloration circuit will have $4\Delta_c$ entangling layers.

The product coloration circuit can also be applied to the LP codes we use in this work. As shown in Extended Data Fig. 1b, a LP code has the same product vertical (horizontal) connectivity as a HGP code when flattening the inner nodes vertically (horizontally). Thus, the same product coloration circuit can be applied to the LP codes with an extra step of flattening the inner codes in between establishing the horizontal and vertical connections.

To further reduce the depth of the syndrome extraction circuit, we also propose a modification of the above circuit in Algorithm 3 and Extended Data Fig. 3, which we refer to as the pipelined product coloration circuit. Here, the main challenge is to choose a gate ordering such that the desired X and Z syndromes are correctly extracted. By performing pipelining and extracting the X syndrome of the second round simultaneously with the Z syndrome of the first round, we can ensure that the gate ordering is always valid while reducing the number of entangling layers required to perform d rounds of syndrome measurement to $(2d+2)\Delta_c$. This could be particularly relevant in further suppressing the effect of idling errors as well as improving the performance of logical gates, which in our scheme require d rounds of repetition. However, our numerical simulations all make use of the product coloration circuit, and we leave the exploration of other syndrome extraction circuits to future work.

Algorithm 2. Product coloration circuit for HGP syndrome extraction

Input : Edge colorations $\mathcal{C}_h, \mathcal{C}_v$ of Tanner graphs associated with horizontal and vertical classical codes C_h, C_v that form the hypergraph product code.
Output : Measurement outcomes of all X and Z stabilizer generators.

`// X stabilizers`
Apply a Hadamard on all data qubits.
Prepare an ancilla in $|+\rangle$ for each X stabilizer and move all X ancilla qubits (red) into the LDPC grid region shown in Fig. Extended Data Figure 1. Do not include any Z ancilla qubits (green).
for *direction* $f \in \{horizontal, vertical\}$ **do**
  for *color* $c \in \mathcal{C}_f$ **do**
    Apply algorithm 1 in direction f, across the whole grid, to bring each pair of qubits connected by an edge of color c in direction f together.
    Apply a CZ gate on each pair of neighboring qubits.
Apply a Hadamard on all data qubits.
Move X ancilla qubits out of the grid region and measure them in the X basis.
`// Z stabilizers`
Prepare an ancilla in $|+\rangle$ for each Z stabilizer and move all Z ancilla qubits (green) into the LDPC grid region shown in Fig. Extended Data Figure 1. Do not include any X ancilla qubits (red).
for *direction* $f \in \{horizontal, vertical\}$ **do**
  for *color* $c \in \mathcal{C}_f$ **do**
    Apply algorithm 1 in direction f, across the whole grid, to bring each pair of qubits connected by an edge of color c in direction f together.
    Apply a CZ gate on each pair of neighboring qubits.
Move Z ancilla qubits out of the grid region and measure them in the X basis.

Algorithm 3. Pipelined product coloration circuit for multi-round HGP syndrome extraction

Input : Edge colorations $\mathcal{C}_h, \mathcal{C}_v$ of Tanner graphs associated with horizontal and vertical classical codes C_h, C_v that form the hypergraph product code.
Input : Number of syndrome repetition rounds d.
Output : Measurement outcomes of all X and Z stabilizer generators.

// X stabilizers of the first round
Apply a Hadamard on data qubits in the bottom left block (orange) of Fig. Extended Data Figure 1.
Prepare an ancilla in $|+\rangle$ for each X stabilizer and move all X ancilla qubits (red) into the LDPC grid region shown in Fig. Extended Data Figure 1. Do not include any Z ancilla qubits (green).
for *color* $c \in \mathcal{C}_h$ **do**
    Apply algorithm 1 in the horizontal direction, across all rows, to bring each pair of qubits connected by an edge of color c in the horizontal direction together.
    Apply a CZ gate on each pair of neighboring qubits.
// Parallel syndrome extraction for $d-1$ rounds
for $i \leftarrow 1$ **to** $d-1$ **do**
    for *direction* $f \in \{vertical, horizontal\}$ **do**
        Apply a Hadamard on all data qubits.
        if $f ==$ *vertical* **then**
            Measure any old Z ancillas in the X basis and prepare a fresh ancilla in $|+\rangle$ for each Z stabilizer.
        else
            Measure any old X ancillas in the X basis and prepare a fresh ancilla in $|+\rangle$ for each X stabilizer.
        Move all X and Z ancilla qubits into their appropriate positions in Fig. Extended Data Figure 1.
        for *color* $c \in \mathcal{C}_f$ **do**
            Apply algorithm 1 in direction f, across all columns, to bring each pair of qubits connected by an edge of color c in direction f together.
            Apply a CZ gate on each pair of neighboring qubits.
// Z stabilizers of the final round
Move X ancilla qubits out of the grid region and measure them in the X basis.
for *color* $c \in \mathcal{C}_v$ **do**
    Apply algorithm 1 in vertical direction, across the whole grid, to bring each pair of qubits connected by an edge of color c in vertical direction together.
    Apply a CZ gate on each pair of neighboring qubits.
Move Z ancilla qubits out of the grid region and measure them in the X basis.

Estimating the rearrangement time and the effect of idling errors

For the schemes described above, we can estimate the amount of time required to implement one round of stabilizer measurements using the technology that has been demonstrated in ref. 22. We assume a transfer time τ_t between a static spatial light modulator trap and a dynamic AOD trap, a peak atom moving acceleration rate of a_p with a cubic spline trajectory, and a uniform grid spacing d. For simplicity, we assume that the number of atoms on a line to be rearranged is a power of 2, $L = 2^k$. The algorithm will, therefore, have a depth of $k = \log_2 L$ layers. To provide enough workspace for shuttling, the total number of traps is $3L/2$.

The compactification step at scale s requires a move of distance at most $sd/2$. Moving all target atoms to the right requires a move of distance at most sd. We can combine the two steps such that we pick up atoms for the next move and drop off atoms from the previous move at the same time; thus, each step requires on average one trap transfer between static and dynamic traps. As described in ref. 22, using a cubic spline movement trajectory, a move of distance l requires time $\sqrt{6l/a_p}$.

The total time required for one layer of full rearrangement before each layer of entangling gates in a syndrome extraction circuit is thus

$$t_{\text{rearrange}} = 2k\tau_t + \sum_{i=0}^{k}\left(\sqrt{\frac{6\times 2^i d}{a_p}} + \sqrt{\frac{3\times 2^i d}{a_p}}\right) < 2\tau_t \log L + (3+2\sqrt{2})\sqrt{\frac{6Ld}{a_p}}. \quad (7)$$

Recent experiments have demonstrated parameters of the order of $\tau_t = 50\,\mu s$, $a_p = 0.02\,\mu m\,\mu s^{-2}$ and $d = 5\,\mu m$. For a moderately sized code consisting of 10,000 qubits (including both data and ancilla qubits), we have $L \approx 100$. The total trap transfer time is 0.7 ms, and the atom movement time is 2.3 ms, for each gate layer. Assuming a (3,4)-biregular graph for the underlying classical expander code, we need eight rounds of rearrangement to measure one full round of stabilizers for Algorithm 3, resulting in a total time overhead of 3 ms per rearrangement layer and 24 ms for a full round of syndrome extraction, a small fraction of the coherence time $T_c > 10$ s that has been demonstrated in neutral-atom arrays[25–27]. This timescale is somewhat longer than the typical readout timescales, and thus, the code cycle time will be dominated by the rearrangement time. The ancilla measurements can be pipelined to happen simultaneously with the atom rearrangements of the following round and, therefore, will not increase the run time. Supplementary Information describes how to perform parallel rearrangement with the LP codes.

The rearrangement time in equation (7) determines the idling errors between sequences of entangling gates in a syndrome extraction circuit. In general, $t_{\text{rearrange}}(n)$ in equation (7) is a function of the code size n, as L is a function of n. Setting the gate error rate p_g as the characteristic physical error rate of our noise model, we rescale the idling error rate $p_i(n)$ together with p_g:

$$p_i(n) = t_{\text{rearrange}}(n)/T_c \times p_g/0.005, \quad (8)$$

where T_c is the atom coherence time and 0.005 is the current CZ gate infidelity demonstrated in ref. 28. Note that the rescaling of $p_i(n)$ with p_g in equation (8) can be justified because idling errors can improve together with the gate errors as hardware improves. For example, further improvements in coherence time can be achieved by detuning trapping light further and better magnetic field shielding. Even without further improvements, coherence times as long as 50 s have been demonstrated in neutral-atom systems[27], sufficient for our analysis at $p_g = 0.1\%$. For all the numerical estimations in this work, we use the upper bound for $t_{\text{rearrange}}(n)$ in equation (7) and use the experimental parameters listed below equation (7), with $T_c = 10$ s.

We numerically verify in Supplementary Information that the effect of the idling errors can be approximated by rescaling the gate errors $p_g \to p_g + 3p_i(n)$ using the product coloration circuit. By replacing the rescaled p_g in the subthreshold scaling for HGP codes in equation (1), we can examine the effect of the idling errors on achievable LFRs. As shown in Extended Data Fig. 4, the LFRs were exponentially suppressed by the code size n when n is small and $3p_i(n) \ll p_g$. For $3p_i(n) > p_g$, they are gradually saturated, then increase and finally approach the gate error threshold. Using the relevant experimental

parameters, the idling errors are negligible for n up to ~10^7 and the LFRs can go below 10^{-24} (green curve), which already suffices for implementing practical quantum algorithms.

Thus, due to the inclusion of idling errors, the LFR cannot be arbitrarily suppressed asymptotically according to the no-go theorems in refs. 15,64. However, constant overhead and fault tolerance can still be achieved at physically relevant sizes by utilizing the quasi-nonlocal connectivity in atom arrays.

Details of teleportation

Here, we describe our teleportation scheme between the qLDPC code and the surface code. We select a logical $\bar{X}_1$ operator for the qLDPC code of minimum weight. This ensures that it contains no sublogicals, which are inequivalent logical operators contained in its support, so that the scheme is fault-tolerant under data errors (see Supplementary Information for a proof). We then associate the qubit support of $\bar{X}_1$ of the qLDPC code ($\bar{Z}_2$ of the surface code) to the bits of a classical code C_1 (C_2) and associate the Z (X) stabilizers of the qLDPC (surface) code with the support on $\bar{X}_1$ ($\bar{Z}_2$) to the checks of C_1 (C_2). Denoting H^1 (H^2) as the check matrix for C_1 (C_2), $H^1_{ij}(H^2_{ij}) = 1$ if the ith Z (X) stabilizer checks the jth qubit of $\bar{X}_1$ (Z_2). We construct an ancilla code patch as a HGP code by taking the HGP of C_1 and C_2. The HGP code encodes a single logical qubit with a logical X and Z representative associated with the bits of C_1 and C_2, respectively.

Lattice surgery between the ancilla patch and the qLDPC (surface) code is realized by merging and splitting along C_1 (C_2), assisted by an extra array of ancillary qubits (Fig. 4a). For a classical linear code C with check matrix H, we denote by C^T its transposed code defined by the check matrix H^T. Taking the qLDPC-ancilla surgery as an example, we insert an extra array of X stabilizers (initialized in $|+\rangle$) and qubits (initialized in $|0\rangle$) in the middle, associated with the checks and the bits of C_1^T, respectively. During the code merging, the Z stabilizers of the qLDPC and the ancilla patch associated with the ith check of C_1 are each modified to include the middle qubit associated with the ith bit of C_1^T. The middle X stabilizer associated with the jth check of C_1^T checks the two qubits of the qLDPC and surface codes associated with the jth bit of C_1 as well as some middle qubits given by the incident relation of C_1^T. It is easy to verify that all the new stabilizers commute as the added and modified qubits and checks across the merged boundary form an HGP code with C_1 and a length-2 repetition code locally. The product of the middle X stabilizers gives the joint logical operator to measure. See Supplementary Information for more algebraic details of the above lattice-surgery scheme and a proof of its fault tolerance under data errors, as well as further details of the numerical simulation.

Note that the lattice-surgery approach in ref. 18 can also be used for teleportation between a qLDPC code and a surface code. Their approach essentially uses an ancilla patch formed by the HGP of a length-d repetition code and a 'union' of C_1 and C_2 associated with the logical operators of the qLDPC code and the surface code, respectively, and directly performs a joint logical measurement on the qLDPC and surface code. The ancilla patch has size $2d^2$ (if using minimum-weight logical operators), which is twice as larger as our ancilla. Compared to their approach, our scheme has a lower space overhead but a larger temporal overhead overall.

Data availability

The data collected and analysed in this article are available at https://doi.org/10.5281/zenodo.8278063 (ref. 65). Source data are provided with this paper.

Code availability

All codes used to generate the figures are available upon request.

References

55. Leverrier, A. & Zémor, G. Quantum Tanner codes. Preprint at https://arxiv.org/abs/2202.13641 (2022).
56. Gu, S., Pattison, C. A. & Tang, E. An efficient decoder for a linear distance quantum LDPC code. In *Proc. Annual ACM Symposium on Theory of Computing* 919–932 (ACM, 2022).
57. Dinur, I., Hsieh, M.-H. H., Lin, T.-C. C. & Vidick, T. Good quantum LDPC codes with linear time decoders. In *Proc. Annual ACM Symposium on Theory of Computing* 905–918 (ACM, 2022).
58. Lin, T.-C. & Hsieh, M.-H. Good quantum LDPC codes with linear time decoder from lossless expanders. Preprint at https://arxiv.org/abs/2203.03581v1 (2022).
59. Richardson, T. & Urbanke, R. *Modern Coding Theory* (Cambridge Univ. Press, 2008).
60. Fawzi, O., Grospellier, A. & Leverrier, A. Efficient decoding of random errors for quantum expander codes. In *Proc. Annual ACM Symposium on Theory of Computing* 878–889 (ACM, 2017).
61. Raveendran, N. et al. Finite rate QLDPC-GKP coding scheme that surpasses the CSS Hamming bound. *Quantum* **6**, 767 (2022).
62. Dordevic, T. et al. Entanglement transport and a nanophotonic interface for atoms in optical tweezers. *Science* **373**, 1511–1514 (2021).
63. Levine, H. et al. Parallel implementation of high-fidelity multiqubit gates with neutral atoms. *Phys. Rev. Lett.* **123**, 170503 (2019).
64. Baspin, N., Fawzi, O. & Shayeghi, A. A lower bound on the overhead of quantum error correction in low dimensions. Preprint at https://arxiv.org/abs/2302.04317v1 (2023).
65. Xu, Q. et al. Constant-overhead fault-tolerant quantum computation with reconfigurable atom arrays. *Zenodo* https://doi.org/10.5281/zenodo.8278063

Acknowledgements

We acknowledge helpful discussions with D. Bandyopadhyay, N. Breuckmann, M. Cain, L. Cohen, C. Duckering, S. Ebadi, S. Evered, X. Gao, S. Geim, M. Kalinowski, S. Li, J. Liu, T. Manovitz, B. Matuz, N. Maskara, Q. Nguyen, H. P. Nautrup, D. Orsucci, N. Rengaswamy, M. Vasmer, P. Yu and H. Zheng, among others. We particularly thank A. Krishna for detailed feedback on our results and paper. We are grateful for the support from the University of Chicago Research Computing Center for assistance with numerical simulations. This work was supported by the Army Research Office (ARO; Grant No. W911NF-23-1-0077 to Q.X. and L.J.), ARO Multidisciplinary University Research Initiatives (MURI; Grant No. W911NF-21-1-0325 to Q.X. and L.J. and Grant No. W911NF-20-1-0082 to J.P.B.A., D.B., M.D.L. and H.Z.), Air Force Office of Scientific Research MURI (Grant Nos. FA9550-19-1-0399 and FA9550-21-1-0209 to Q.X. and L.J. and Grant No. FA9550-19-1-0360 to C.A.P.), the National Science Foundation (NSF; Grant Nos. OMA-1936118, ERC-1941583 and OMA-2137642 to Q.X. and L.J. and Grant Nos. CCF-2313084, CIF-1855879, CCF-2100013 and CIF-2106189 to N.R. and B.V.), NTT Research (L.J.), the Packard Foundation (Grant No. 2020-71479 to L.J.), the Quantum Systems Accelerator Center of the US Department of Energy (Grant Nos. 7568717 and DE-SC0021013 to J.P.B.A., D.B., M.D.L. and H.Z.), the Center for Ultracold Atoms (Grant No. NSF PHY-1734011 to J.P.B.A., D.B., M.D.L. and H.Z.), the NSF Institute for Quantum Information and Matter (C.A.P.), the Optimization with Noisy Intermediate-Scale Quantum Devices programme of the Defense Advanced Research Projects Agency (Grant No. W911NF2010021 to D.B., J.W., M.D.L. and H.Z.), the NSF Graduate Research Fellowship Program (Grant No. DGE1745303 to D.B.), the Fannie and John Hertz Foundation (D.B.), and the Ramsay Centre for Western Civilisation (J.P.B.A.).

Author contributions

M.D.L. and L.J. conceived the project. Q.X., J.P.B.A. and C.A.P. performed the numerical simulations. Q.X., J.P.B.A., C.A.P. and H.Z. proved the fault tolerance of the schemes. H.Z. identified the correspondence between product codes and product optical tools, H.Z. and Q.X. devised efficient implementations of various codes, and D.B. verified the experimental feasibility of the proposal. J.W. and H.Z. devised the 1D log-depth rearrangement algorithm. N.R. constructed the LP codes used in the

simulations. All authors contributed to the design of the methodology and data analysis. All authors contributed to the writing of the paper. All work was supervised by B.V., M.D.L., L.J. and H.Z.

Competing interests

M.D.L. is a co-founder and shareholder of QuEra Computing. J.W and H.Z. are employees of QuEra Computing. The other authors declare no competing interests.

Additional information

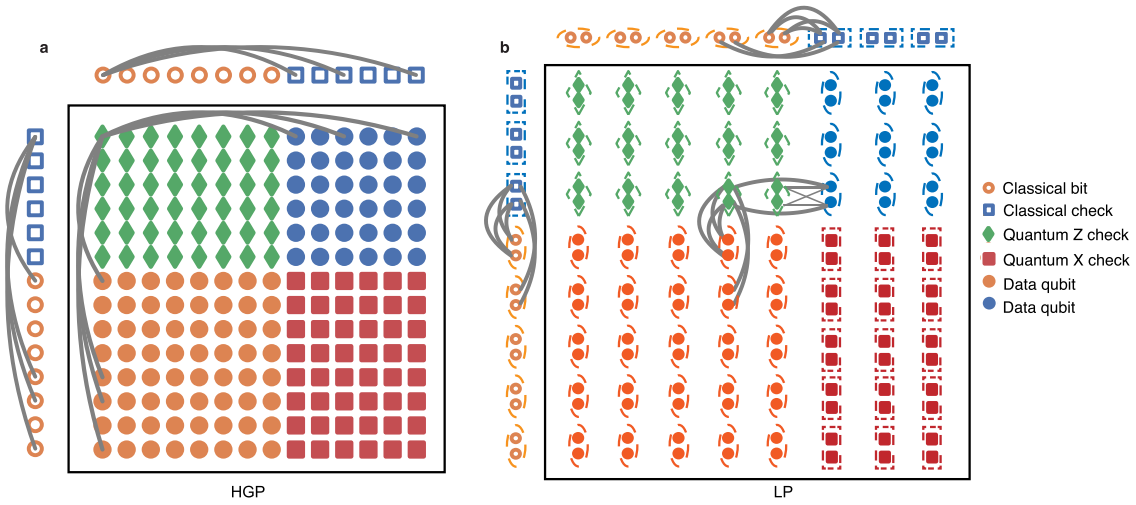
Extended data is available for this paper at https://doi.org/10.1038/s41567-024-02479-z.

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41567-024-02479-z.

Correspondence and requests for materials should be addressed to Liang Jiang or Hengyun Zhou.

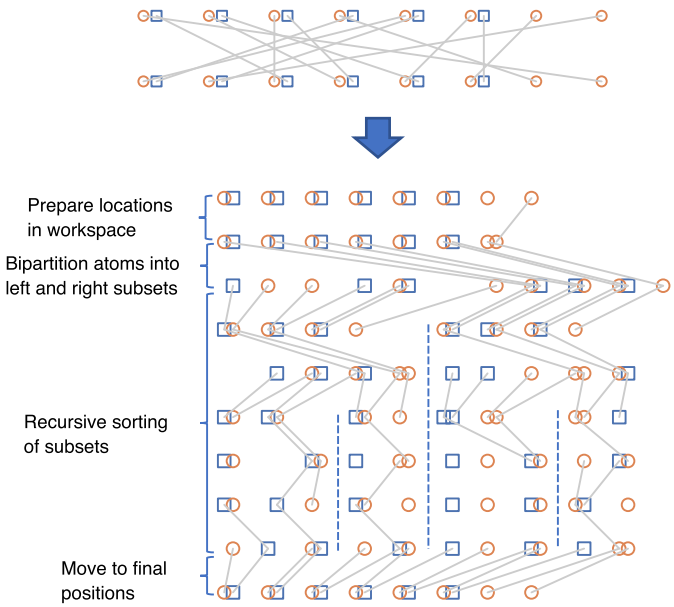
Peer review information *Nature Physics* thanks the anonymous reviewer(s) for their contribution to the peer review of this work.

https://doi.org/10.1038/s41567-024-02479-z



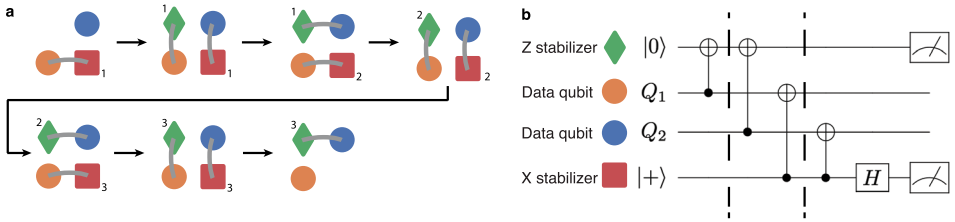
Extended Data Fig. 1 | Product structure of HGP codes and LP codes. (**a**) The HGP code is constructed from two classical LDPC codes. The classical codes are illustrated on the left and top, where circles indicate classical bits and squares indicate classical checks. A data qubit is placed at each intersection of two classical bits (filled orange circles) and of two classical checks (filled blue circles). *Z* stabilizer generators are placed at the intersection of horizontal bits and vertical checks, while *X* stabilizer generators are placed at the intersection of horizontal checks and vertical bits. Each stabilizer is connected to data qubits along the same row or column, with the same connectivity as the classical codes, as illustrated for the top left *Z* stabilizer. We have omitted other connections for ease of visualization. (**b**) The LP code is constructed by taking a lift over the hypergraph product of two classical protographs. The protographs and their hypergraph product are indicated by the dashed nodes and the lift is illustrated by the multiple inner nodes within each dashed node. The inner connectivity between two dashed nodes is given by the circulant-matrix representation of the ring elements in Eq. (4). When flattening the inner nodes vertically (horizontally), the vertical (horizontal) connectivity between the qubits and the checks for each column (row) is the same as the left (top) lifted classical code.

https://doi.org/10.1038/s41567-024-02479-z



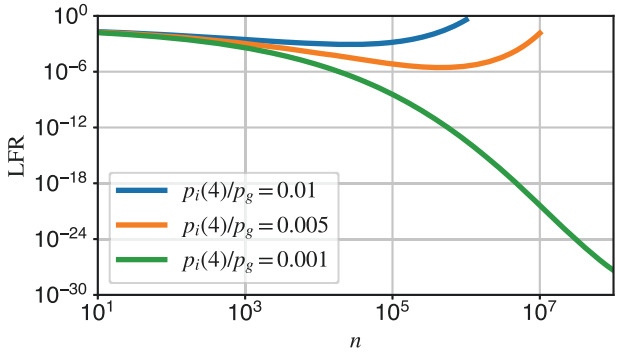
Extended Data Fig. 2 | Efficient non-intersecting rearrangement in log-depth. By using a divide and conquer algorithm, we can perform an arbitrary 1D rearrangement in depth logarithmic in the number of qubits. Repeating this across the array yields an efficient implementation of the desired rearrangements, without requiring intersecting atom trajectories that may lead to additional loss and decoherence. Here, we illustrate the full set of movements required in a small example. Similar to the earlier figures, blue squares indicate classical checks and orange circles indicate classical bits. When a blue square and orange circle are moved to be neighboring at the end of the rearrangement, they execute an entangling gate. The top panel indicates the desired change of configuration, where the ordering of neighboring atoms in the top row needs to be modified to that in the bottom row via parallel rearrangement, as illustrated by the crossing gray lines. The bottom figure illustrates how we decompose the arbitrary rearrangement into a non-crossing rearrangement, where the gray lines no longer intersect.

https://doi.org/10.1038/s41567-024-02479-z



Extended Data Fig. 3 | Illustration of ordering of operations in pipelined syndrome extraction. (**a**) Successive steps of entangling gates for the pipelined product coloration circuit described in Alg. 3, with $d = 3$ rounds of syndrome extraction. Numbers at the corners of the X and Z ancilla qubits denote the round of syndrome extraction they correspond to. (**b**) Illustration of a local circuit that data qubits and ancilla qubits see, with dashed lines indicating different circuit moments. As the X stabilizer interacts with both qubits before the Z stabilizer, the syndrome extraction order is valid. Similar analysis can be performed for the commutation relations with the next round of ancilla qubits.

https://doi.org/10.1038/s41567-024-02479-z



Extended Data Fig. 4 | Achievable logical failure rates of the HGP codes with different idling error strengths. We characterize the idling error strengths as the relative ratio between the idling error rate $p_i(n)$ at $n = 4$ and the gate error rate p_g. This idling error strength can potentially be reduced by, for example, increasing the coherence time and accelerating the atom shuttling.