Access Improvements to Densely Packed Quantum Memory

Kabir Dubey* and Kaitlin N. Smith
Department of Computer Science, Northwestern University
Evanston, IL, USA
**kabir@u.northwestern.edu*

***Abstract*—Fault-tolerant quantum memory is essential for large-scale quantum computer systems and has recently achieved major experimental and theoretical advances. In 2023, McEwen, Bacon, and Gidney showed that walking code circuits move surface code logical qubits diagonally while maintaining comparable logical performance to standard surface code circuits. Building on this work, we apply gliding codes to create access hallways in densely packed qubit arrays using minimal ancilla space. This approach provides arbitrary access to stored qubits and supports cache-like eviction of qubits from the storage array. For a storage layout of $l \times w$ surface code logical qubits, our design reduces spacetime volume by $O(lw)$ when compared to loose packings which allocate rows and columns of ancilla qubit patches between each logical qubit.**

I. Introduction

Quantum computer systems typically use hot and cold storage units for resource management at scale. In error-protected storage, high-quality logical qubits are actively maintained while computational resources are generated on demand. The distinction between hot and cold depends on the access latency for the operation of interest. Such a design offers quantum computer systems enhanced resource allocation capabilities and greater resilience to errors.

For instance, when connectivity is limited, one may maximize the use of a quantum LDPC code by concatenating it with an outer code that has a higher ratio of logical qubits to physical qubits [6]. The surface code is well-suited for this technique, as it provides high-quality logical qubits that can be assigned to storage units with increased protection from parity checks along rows and/or columns called yokes. Yoked surface codes singlehandedly nearly triple the logical encoding rate of standard surface codes in the teraquop regime [7]. 2D yoked logical qubits in cold storage reach a logical error rate of 10^{-15} with 430 physical qubits per logical qubit. Alongside other techniques, this reduces the estimated spatial footprint of factoring a 2048 bit RSA integer by an order of magnitude [5]. Dynamical codes in general are expected to rely on similar storage concepts for their system operations, provided that a compiler can efficiently schedule operations between subsystems.

The topology of a quantum computer architecture completely determines its compilation model for resolving two-qubit interactions, which itself enables the movement of logical information throughout the computer system. In the context of topological quantum error correction (TQEC), these movement operations are performed using lattice surgery, requiring either ancillary hallways between and/or around each logical qubit or a compact allocation computed heuristically. Here, we consider techniques to organize storage units with low spacetime volume using walking surface code circuits.

II. Walking Codes

An eye for the overall spacetime structure of topological codes, as opposed to static stabilizer circuits, informs time-dynamical surface code circuits which achieve comparable logical performance to standard surface codes while relaxing hardware requirements [1], [8]. In the time-dynamic picture, one may analyze code stabilizers by aggregating the parity checks done by the measurements in every cycle circuit through space and time. A *walking surface code circuit* leverages the freedom provided by allowing detecting region shapes to break the fixed location of code stabilizers on the underlying physical qubit grid. The circuit is comprised of *stepping circuits* which are identical to a standard surface code cycle circuit except the last layer of CNOT gates reverse the control and target of the first CNOT layer (Fig. 12, [8]). Degrees of freedom for qubit reset and reuse allow walking surface code instructions to replace lattice surgery instructions in a way that reduces space without increasing time or reducing the code distance. Movement patterns enabled by walking encompass oscillations where the patch returns to its initial position every two cycles (*wiggling*), continuous unidirectional diagonal movement (*gliding*), and lateral displacement achieved though alternating rectilinear steps (*sliding*).

Gliding enables surface code patch movement in $2d$ rounds. Rather than taking a half-step in each round, one can flip the direction of the first and last CNOTs in the cycle, as opposed to just the last, and reduce the movement time to d rounds [4]. Gliding reduces the path length of a movement from the origin to a point (i, j) to $\min(|i|, |j|)$ compared to the rectilinear cost of $|i| + |j|$ for lattice surgery. Appendix C of [8] shows that gliding codes can shift a dense linear register of N logical qubit patches by S qubits in a diagonal manner, saving $N^2 - 2NS$ units of spacetime volume compared to lattice surgery.

III. Hot storage in a dense packing

We expand the shift procedure in [8] by considering dynamic gliding instructions that temporarily open ancilla hallways. This enables flexible, low-latency qubit communication channels within an otherwise dense hot storage layout. Our storage design accommodates an array of $l \times w$ rotated surface code logical qubits, each with code distance d. An external source qubit u must traverse a path P through the storage area

to reach a sink position v for interaction with a target qubit in storage, all while maintaining logical error protection. A conventional hot storage area with both row and column ancilla hallways (Fig. 1, left) would require $((2l+1)(2w+1)-lw)d^2$ qubits of ancilla space, allowing the source qubit to enter or exit the storage unit in d rounds via lattice surgery.

Dense packing offers an efficient alternative if two adjacent borders of ancilla qubits are allocated to enable any subset of the array to slide or glide in and out (Fig. 1, right). This approach reduces the ancilla space to $(l + w + 1)d^2$ qubits, though it introduces additional time overhead in the form of hallway creation. Moving all logical qubits obstructing path P in parallel along a single-file diagonal line requires $2d$ rounds to create the necessary ancilla hallway. The source qubit can then traverse the cleared path in d rounds, yielding a total movement time of $3d$ rounds.

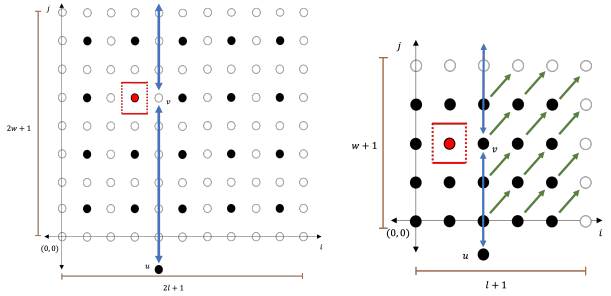

Fig. 1: Storage of $l = 5$ by $w = 4$ logical qubits, with the $u \to v$ path in blue. Loose packing is left, dense packing is right. Diagonal gliding in green opens the ancillary hallway. Each of the two red dashed (resp. solid) edges represent the target qubit's X (resp. Z) boundary operator.

To compare the strategies quantitatively, we use the spacetime product to get the cost $C^{(\text{loose})} \coloneqq (2l+1)(2w+1)d^2 \cdot d = (4lw + 2l + 2w + 1)d^3$ for lattice surgery in a loose packing, and $C^{(\text{dense})} \coloneqq (l+1)(w+1)d^2 \cdot (d+2d) = 3(lw+l+w+1)d^3$ for gliding in our dense packing. The difference, in units of spacetime blocks, is given by,

$$\Delta C \coloneqq (C^{(\text{loose})} - C^{(\text{dense})})/d^3 = lw - l - w - 2, \quad (1)$$

which is a monotonic increasing and nonnegative function for $l = 2$ (resp. $w = 2$) and $w \geq 4$ (resp. $l \geq 4$) or both $l, w \geq 3$.

We assume that exposing only one boundary of the target qubit suffices for any packing allocation and access step. Compared to packings with only ancilla columns or rows, we lose the spacetime volume advantage if gliding requires one step per cycle, but maintain it for rectangular packings which glide at two steps per cycle. To expose the opposite boundary, one may either: (a) perform a patch rotation on the target qubit [7], (b) glide both the target and all qubits in the column above it, or (c) allocate a full *perimeter* of ancilla space to glide the *rows* below the target.

An additional complication arises from the fact that opening access hallways with gliding has each qubit incur an error suppression penalty because walking circuit performance is not as good as static circuits [1], [8]. Therefore, we expect access policies involving lattice surgery or sliding circuits (optionally synchronized with wiggling) to potentially decrease overhead by moving less qubits overall. We defer the simulation of the overhead associated with the aforementioned strategies and their combinations to future work.

IV. Discussion

Under assumptions of parallel control and agnostic weighting of space and time resources, our layout achieves strictly better resource efficiency than loose packings for any nontrivial storage area. Our approach does not offer better access latency than loose packing or a smaller spatial footprint than dense packing, but can instead be viewed as a *warm storage* design that provides the minimal spatial overhead needed for moderate access latency. Since most TQEC systems must be decoded in real-time with minimal congestion and blocking, we expect many compiler workloads will benefit from a cache hierarchy based on access latency.

The viability of our procedure remains to be fully established because we have not yet characterized the logical error rate of our instruction set. While one may extrapolate the performance of gliding codes from Fig. 17 in [8], definitive assessment requires dedicated simulation. Moreover, although large-scale circuit simulations can be executed in `Stim` through parallelized jobs using `sinter`, such analyses are seldom undertaken due to the difficulty of building large circuits [3], [7]. To address this limitation, we employ the open-source software package `tqec`, which models logical operations as (2+1)-dimensional blocks and compiles them into `Stim` files for generating plots that confirm error suppression at the expected scaling $p^{(d+1)/2}$ for physical qubit error rate p [2]. We are currently developing functionality within `tqec` to represent storage units and walking surface code circuits, with results reserved for future publication.

V. Acknowledgments

We thank Matt McEwen for helpful comments. We thank the `tqec` Design Automation community for being a space to learn, discuss, and test the ideas presented in this paper [2]. This research was supported by Google's Research Scholar Program.

References

[1] Google Quantum AI et al. Demonstrating dynamic surface codes. *arXiv preprint arXiv:2412.14360*, 2024.
[2] TQEC Community. Design automation for topological quantum error correction, publication pending. tqec.github.io/tqec/, 2025.
[3] Craig Gidney. Stim: a fast stabilizer circuit simulator.
[4] Craig Gidney. Inplace access to the surface code y basis. *Quantum*, 8:1310, April 2024.
[5] Craig Gidney. How to factor 2048 bit rsa integers with less than a million noisy qubits, 2025.
[6] Craig Gidney and Thiago Bergamaschi. A constant rate quantum computer on a line, 2025.
[7] Craig Gidney, Michael Newman, Peter Brooks, and Cody Jones. Yoked surface codes, 2023.
[8] Matt McEwen, Dave Bacon, and Craig Gidney. Relaxing Hardware Requirements for Surface Code Circuits using Time-dynamics. *Quantum*, 7:1172, November 2023.