tqec: A Python package for topological quantum error correction

Adrien Suau [1], **Yiming Zhang** [2], **Purva Thakre**[3], **Yilun Zhao** [4], **Kabir Dubey** [5], **Jose A Bolanos** [6], **Arabella Schelpe** [7], **Tianyi Hao** [8], **Philip Seitz** [9], **Gian Giacomo Guerreschi**[10], **Ángela Elisa Álvarez Pérez** [11], **Reinhard Stahn** [12], **Jerome Lenssen** [13], **Brendan Reid**[14], **and Austin Fowler** [15]

1 Qraftware, Toulouse, France **2** University of Science and Technology of China, China **3** School of Physics and Applied Physics, Southern Illinois University, Carbondale, USA **4** Institute of Computing Technology, Chinese Academy of Sciences, China **5** Department of Computer Science, Northwestern University, USA **6** Independent Consultant, Finland **7** Independent Researcher, UK **8** University of Wisconsin-Madison, USA **9** Technical University of Munich, TUM School of Computation, Information and Technology, Germany **10** Intel Corporation, Technology Research Group, Santa Clara, USA **11** Solvy, Spain **12** Parity Quantum Computing Germany GmbH, Hamburg, Germany **13** VTT, Finland **14** PsiQuantum, Palo Alto, California, USA **15** Stairway Invest, Los Angeles, California, USA

DOI: 10.21105/joss.09142

Software

- Review
- Repository
- Archive

Editor: Owen Lockwood

Reviewers:

- @burgholzer
- @meandmytram

Submitted: 04 September 2025
Published: 01 April 2026

Summary

tqec is a Python-based open-source compiler that takes a logical-level quantum computation model represented as connected 3D primitive blocks and translates it into a detailed, fault-tolerant, physical-level circuit. The result is a Stim circuit (Gidney, 2021) with all the detailed information needed for simulation or to run on real quantum hardware. This enables both quantum algorithm designers and experimentalists to rapidly iterate and obtain exact low-level circuits, facilitating efficient performance simulation or experimental demonstration. At present, tqec is primarily centered on the surface code.

Statement of Need

Simulations of quantum computer operations in the large-scale error correction regime are currently infeasible. As a result, the practical performance of fault-tolerant computations cannot be evaluated accurately. Building the logical Stim circuits required for error-corrected computation is a complex and time consuming process, and Monte Carlo simulations at the scale of, for example, hierarchical memory systems involving yoked surface codes, are difficult to perform exactly (Gidney et al., 2025). By automating the compilation from high-level logical circuits defining lattice surgery protocols to Stim circuits, tqec enables exact Stim-based simulation and replaces the need for empirical extrapolations in fault-tolerant resource estimation (Gidney, 2021) .

tqec is designed to be used by students and researchers who seek to understand the theory of quantum error correction and experiment with scalable quantum computer system and circuit designs. A poster featuring preliminary research and an educational tutorial enabled by tqec have been approved for conference proceedings: Dubey & Smith (2025) and Kan et al. (2025). A further software package has been recently built to enable better interfacing between PyZX and tqec: Bolanos & Fowler (2025). The functionality of the tqec package is based on several academic papers (A. G. Fowler et al., 2012; Gidney et al., 2025; Kissinger & van de Wetering, 2020; McEwen et al., 2023; Polian & Fowler, 2015), and makes substantial use of Craig Gidney's Stim package (Gidney, 2021).

State of the Field

The tqec library emerged from Austin Fowler's call-to-action presentation at the Munich Quantum Software Forum (A. Fowler, 2023) which advocated for an open-source collaborative effort to build software for quantum error correction (QEC). Several software libraries have been released publicly to attempt to tackle the various challenges related to fault-tolerant compilation. Of the compiler libraries discussed in this section, tqec stands out as uniquely positioned to tackle these obstacles. Where many alternatives offer limited functionality or have fallen into disrepair, tqec is actively developed and supported by a thriving community.

To our knowledge, the Lattice Surgery Compiler by Watkins et al. (2024b) was the first publicly released software to compile a QASM circuit into lattice surgery operations based on the surface code. While active development on this project has ceased (Watkins et al., 2024a), an upgraded version of the compiler was released (Leblond et al., 2024) to enable hardware aware, resource optimized, DAG-based parallel compilation of lattice surgery instructions for the Clifford + T gate set circuits. Robertson et al. (2025) introduced another surface code lattice surgery compiler that factors in resource estimation to compile quantum computations fault-tolerantly building on the approach presented in Litinski (2019). This software extends beyond tqec by incorporating logical qubit mapping, routing, and allocation; each a critical component of a fully automated compilation pipeline. All three projects employ their own native intermediate representation and gate-level compilation strategies tailored to their research goals, limiting their flexibility. Unlike tqec, these tools do not output Stim circuits, which are essential for gauging the performance of Clifford computations before deploying to physical hardware. tqec directly represents lattice surgery via its native BlockGraph data structure, enabling both manual and automated optimization. Introducing hardware aware compilation capabilities is on the tqec roadmap and will be addressed in the future.

Substrate Scheduler by Liu et al. (2023) compiles fault tolerant graph states based on the formalism in Litinski (2019) weighing the tradeoffs between the speed of the computation and qubit overhead in surface code patches. Substrate Scheduler was designed with the goal to minimize the space-time volume of the generated fault-tolerant computation. It is limited to fault-tolerant compilation of graph states and is no longer under active development.

Loom Design by Entropica Labs (2025) is a software project designed to evaluate the performance of QEC protocols in general. The project contains a built-in library of QEC codes (color codes, surface codes, rotated surface codes, etc.) to implement end-to-end lattice surgery protocols. While tqec utilizes multiple spatial junction types and stretched stabilizers for hook error handling in surface codes, Loom Design is limited by a generic surface code layout. Compared to the Loom Design 3D visualizer, tqec also provides support for a comprehensive range of 3D structures enabling automated correlation surface finding.

Acknowledgements

We thank the Unitary Foundation for a micro-grant in the early stages of the project development.

References

Bolanos, J. A., & Fowler, A. G. (2025). *Topologiq: Algorithmic lattice surgery*. https://github.com/jbolns/topologiq

Dubey, K., & Smith, K. N. (2025). Access improvements to densely-packed quantum memory. *2025 IEEE International Conference on Quantum Computing and Engineering*. https:

//doi.org/10.1109/qce65121.2025.10483

Entropica Labs. (2025). *Loom: Python library for the simplified setup of quantum error correction codes*. https://github.com/entropicalabs/el-loom

Fowler, A. (2023). *Computing with fewer qubits: pitfalls and tools to keep you safe*. Munich Quantum Software Forum. https://www.youtube.com/watch?v=aUtH7wdwBAM&t=6s

Fowler, A. G., Mariantoni, M., Martinis, J. M., & Cleland, A. N. (2012). Surface codes: Towards practical large-scale quantum computation. *Phys. Rev. A*, *86*(3), 32324. https://doi.org/10.1103/PhysRevA.86.032324

Gidney, C. (2021). Stim: A fast stabilizer circuit simulator. *Quantum*, *5*(497). https://doi.org/10.22331/q-2021-07-06-497

Gidney, C., Newman, M., Brooks, P., & Jones, C. (2025). Yoked surface codes. *Nature Communications*, *16*(4498). https://doi.org/10.1038/s41467-025-59714-1

Kan, S., Thakre, P., Dubey, K., Suau, A., Zhang, Y., Fowler, A., Li, A., Stein, S., & Mao, Y. (2025). TUT 12 – automated topological quantum error correction using 3D primitives. *2025 IEEE International Conference on Quantum Computing and Engineering*. https://qce.quantum.ieee.org/2025/tutorials-abstracts/

Kissinger, A., & van de Wetering, J. (2020). PyZX: Large scale automated diagrammatic reasoning. *Proceedings 16th International Conference on Quantum Physics and Logic*. https://doi.org/10.4204/EPTCS.318.14

Leblond, T., Dean, C., Watkins, G., & Bennink, R. (2024). Realistic cost to execute practical quantum circuits using direct Clifford+T lattice surgery compilation. *ACM Transactions on Quantum Computing*, *5*(4), 1–28. https://doi.org/10.1145/3689826

Litinski, D. (2019). A game of surface codes: Large-scale quantum computing with lattice surgery. *Quantum*, *3*, 128. https://doi.org/10.22331/q-2019-03-05-128

Liu, S., Benchasattabuse, N., & Morawiec, S. (2023). *Substrate scheduler: A compiler to generate fault-tolerant graph states using the stabilizer formalism*. https://github.com/sfc-aqua/gosc-graph-state-generation

McEwen, M., Bacon, D., & Gidney, C. (2023). Relaxing hardware requirements for surface code circuits using time-dynamics. *Quantum*, *7*(1172). https://doi.org/10.22331/q-2023-11-07-1172

Polian, I., & Fowler, A. G. (2015). Design automation challenges for scalable quantum architectures. *52nd ACM/EDAC/IEEE Design Automation Conference (DAC), 1–6*. https://doi.org/10.1145/2744769.2747921

Robertson, A., Gao, H., & Sanders, Y. R. (2025). *A resource allocating compiler for lattice surgery*. https://arxiv.org/abs/2506.04620

Watkins, G., Nguyen, H. M., Watkins, K., Pearce, S., Lau, H.-K., & Paler, A. (2024a). *Lattice surgery compiler: Lattice surgery quantum error correction compiler*. https://github.com/latticesurgery-com/lattice-surgery-compiler

Watkins, G., Nguyen, H. M., Watkins, K., Pearce, S., Lau, H.-K., & Paler, A. (2024b). A high performance compiler for very large scale surface code computations. *Quantum*, *8*, 1354. https://doi.org/10.22331/q-2024-05-22-1354