

¹ LogicalQ: A toolkit for quantum circuit development ² with generalized quantum error mitigation, detection, ³ and correction

⁴ Rasmit Devkota  ^{1,2}, Ben Hagan³, Nolan Heffner  ^{1,2}, Younos Hashem  ¹,
⁵ Noah Song  ², Arhan Deshpande¹, Fiyin Makinde⁴, Richard Yu^{2,5}, Alisa
⁶ Petrusinskaia  ⁵, Zhixin Song¹, David Lloyd George⁶, and Lance Lampert⁶

⁷ School of Physics, College of Sciences, Georgia Institute of Technology, Atlanta, GA ² School of
⁸ Mathematics, College of Science, Georgia Institute of Technology, Atlanta, GA ³ Ming Hsieh Department
⁹ of Electrical and Computer Engineering, Viterbi School of Engineering, University of Southern California,
¹⁰ Los Angeles, CA ⁴ School of Electrical and Computer Engineering, College of Engineering, Georgia
¹¹ Institute of Technology, Atlanta, GA ⁵ School of Computer Science, College of Computing, Georgia
¹² Institute of Technology, Atlanta, GA ⁶ Department of Physics, Duke University, Durham, NC

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: 

Submitted: 26 October 2025

Published: unpublished

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#))

¹³ Summary

¹⁴ LogicalQ is a Python toolkit for quantum circuit development with generalized quantum
¹⁵ error mitigation, detection, and correction (QEMDAC). LogicalQ inherits many of its data
¹⁶ structures from, and thus is designed to interface well with, the Qiskit and Qiskit Aer
¹⁷ packages ([Javadi-Abhari et al., 2024](#)).

¹⁸ The source code for LogicalQ is available on [GitHub](#). It can be installed via pip from the
¹⁹ [pypi index](#). Its [documentation](#) is hosted publicly.

⁵ Statement of need

²¹ Quantum computing presents a new model for computation which may significantly accelerate
²² discovery in many fields, from physics ([Feynman, 1982](#)) to cryptography ([P. W. Shor, 1994](#))
²³ to economics ([Herman et al., 2023](#)). However, the current era of quantum hardware is still
²⁴ noisy, and quantum error mitigation (QEM) ([Viola & Lloyd, 1998](#)), quantum error detection
²⁵ (QED) ([Leung et al., 1999](#)), and quantum error correction (QEC) ([Peter W. Shor, 1995](#))
²⁶ techniques are necessary for the execution of utility-scale algorithms with any reasonable fidelity.
²⁷ Although there exist a number of libraries which allow researchers with some background to
²⁸ run experiments focused on QEC, there is no unified framework for general QEMDAC which is
²⁹ accessible to quantum computing researchers of any background and simultaneously supports
³⁰ the convenient implementation of quantum algorithms.

³¹ Many of the necessary components for QEMDAC have been formalized mathematically such
³² that algorithms can be designed to construct these components for general classes of error
³³ control techniques ([Gottesman, 1997](#)). LogicalQ, like many existing QEMDAC libraries, uses
³⁴ such generalized constructions to meet any use case and application.

³⁵ A comparison of existing libraries is made in Table 1. We choose to compare features which
³⁶ may be desirable to researchers in quantum algorithms. Note that we define external two-way
³⁷ interoperability to be with any external general-purpose quantum computing tool such as QASM
³⁸ or Qiskit, but not just another QEMDAC tool.

Table 1: Comparison of LogicalQ with other major QEMDAC packages; stim is due to (Gidney, 2021), mqt-qecc is due to (Wille et al., 2024), PECOS is due to (Ryan-Anderson, n.d.), stac is due to (Khalid, 2024), and tqec is due to (tqec, n.d.).

Feature	LogicalQ	stim	mqt-qecc	PECOS	stac	tqec
Stabilizer code QEC	✓	✓	✓	✓	✓	✓
qLDPC-oriented QEC	✓	✗	✓	✗	✗	✗
Arbitrary logical Clifford gates	✓	✗	✗	✓	✓	✗
Arbitrary logical non-Clifford gates	✓	✗	✗	✓	✓	✗
Advanced decoders	✗	✓	✓	✓	✗	✓
Arbitrary noise model support	✓	✗	✗	✓	✗	✓
Optimized QEC cycle scheduling	✓	✗	✗	✗	✗	✗
Experiment suite	✓	✗	✗	✓	✗	✓
Logical state analysis	✓	✗	✗	✗	✗	✓
External two-way interoperability	✓	✓	✗	✗	✓	✗
Cloud hardware interfaces	✓	✗	✗	✗	✗	✗

³⁹ In summary, many of the existing libraries are notable for their high-performance simulations
⁴⁰ and advanced implementations of certain features, but none support the full functionality
⁴¹ required for QEMDAC applied to quantum algorithms research, especially on cloud hardware.
⁴² LogicalQ is also unique in that it has a suite of experiments for testing QEMDAC which serves
⁴³ as a quick set of tests for researchers studying noise control.

⁴⁴ LogicalQ was designed to accelerate the application of QEMDAC in quantum algorithm
⁴⁵ development, so its core design principle is maximizing user capability for implementing
⁴⁶ complex quantum circuits and using QEMDAC. The combination of generalized quantum error
⁴⁷ correction functionality, compatibility with libraries such as Qiskit, existence of numerous demo
⁴⁸ notebooks, and overall usability will increase accessibility to quantum error-corrected research
⁴⁹ and enable deeper study into the application of quantum error correction.

⁵⁰ Furthermore, QEMDAC techniques can make analysis of quantum computation results difficult
⁵¹ because they utilize overhead resources which exponentially increase the size of experiment
⁵² outputs. There is a need for tools which can parse QEMDAC results without requiring
⁵³ researchers to understand the often-complex mathematics of these techniques.

⁵⁴ Although many of the necessary tools are not particularly lengthy or convoluted in their
⁵⁵ implementation, LogicalQ provides a single toolkit which handles the complexities of the
⁵⁶ QEMDAC workflow to avoid user error when constructing circuit components or performing
⁵⁷ mathematical analyses.

58 Functionality

⁵⁹ LogicalQ consists of various modules which provide functionality for the construction of
⁶⁰ QEMDAC components as well as their application and analysis.

⁶¹ A general flowchart of library structure is shown in Figure 1.

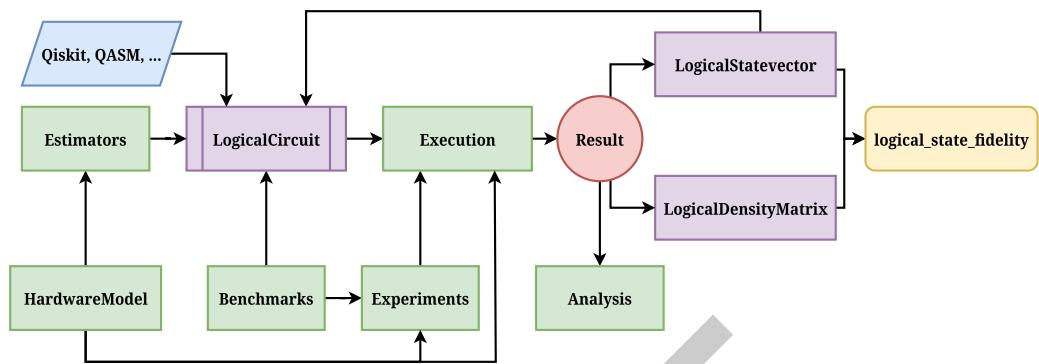


Figure 1: LogicalQ Architecture

- 62 The Logical module lies at the heart of the library with the `LogicalCircuit` class, which
 63 inherits from the `QuantumCircuit` class in Qiskit and extends it with a variety of QEMDAC
 64 features. A `LogicalCircuit` can be constructed from a Qiskit `QuantumCircuit` via the
 65 `from_physical_circuit` method, which enables easy integration of LogicalQ into existing
 66 workflows. The `optimize_qec_cycle_indices` method of `LogicalCircuit` performs cost
 67 accounting based on a constraint model and effective threshold in order to construct an optimal
 68 list of QEC cycle indices.
- 69 The Logical module also contains the `LogicalStatevector` and `LogicalDensityMatrix`
 70 classes, which inherit from the `Statevector` and `DensityMatrix` classes in Qiskit respectively
 71 and enable representation and analysis of quantum states at either the logical level or physical
 72 level. Logical also contains the `logical_state_fidelity` function which is designed to
 73 support mixed-type fidelity comparisons, such as the fidelity of a physical density matrix and a
 74 logical statevector.
- 75 The Benchmarks module contains constructors for many of the most commonly-used bench-
 76 marking circuits in quantum computation, including randomized benchmarking and quantum
 77 volume. These functions expose parameters such as qubit counts, circuit lengths, and random
 78 selection seeds to the user so that they can be directly integrated into controlled tests and
 79 experiments.
- 80 The Experiments module contains a variety of experiments which can be used to study
 81 QEMDAC techniques. Experiment data can be analyzed with functions from the Analysis
 82 module.
- 83 The Execution module contains the `execute_circuits` function, which provides a single
 84 interface for both simulator and hardware backends with smart handling of complex aspects
 85 such as backend communication, hardware models, and transpilation.
- 86 The Estimators module contains special experiments which are used in QED and QEC cycle
 87 scheduling. In particular, this includes effective threshold estimation and constraint model
 88 construction.
- 89 The Library modules contain utilities such as quantum codes for QED and QEC, hardware mod-
 90 els for modelling quantum devices, special gates for benchmarking, and dynamical decoupling
 91 sequences for QEM.

92 Scholarly Work

- 93 LogicalQ development has largely been driven by an ongoing research project to optimize the
 94 scheduling of QEMDAC components in quantum circuits, with the motivation of performing
 95 fault-tolerant Hamiltonian simulations of lattice gauge theories and other physical models on

96 quantum hardware. This involves code switching between QEC and QED codes depending on
97 the error-criticality of a part of a circuit, made less complex by LogicalQ's generalized framework
98 for stabilizer codes. There is also ongoing work on genetic algorithm-based optimization of
99 physical and logical dynamical decoupling sequences for these applications and others in science.

100 Acknowledgements

101 We acknowledge contributions from the Quantum Computing Association at Georgia Tech.

102 We would like to thank Jeff Young for serving as the advisor of the Quantum Computing
103 Association at Georgia Tech and this project.

104 This research was supported in part through research cyberinfrastructure resources and services
105 provided by the Partnership for an Advanced Computing Environment (PACE) at the Georgia
106 Institute of Technology, Atlanta, Georgia, USA ([PACE, 2017](#)).

107 References

108 Feynman, R. P. (1982). Simulating physics with computers. *International Journal of Theoretical
109 Physics*, 21(6–7), 467–488. <https://doi.org/10.1007/bf02650179>

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

111 Gottesman, D. (1997). *Stabilizer codes and quantum error correction*. arXiv. <https://doi.org/10.48550/ARXIV.QUANT-PH/9705052>

112 Herman, D., Googin, C., Liu, X., Sun, Y., Galda, A., Safro, I., Pistoia, M., & Alexeev,
113 Y. (2023). Quantum computing for finance. *Nature Reviews Physics*, 5(8), 450–465.
<https://doi.org/10.1038/s42254-023-00603-1>

114 Javadi-Abhari, A., Treinish, M., Krsulich, K., Wood, C. J., Lishman, J., Gacon, J., Martiel, S.,
115 Nation, P. D., Bishop, L. S., Cross, A. W., Johnson, B. R., & Gambetta, J. M. (2024).
116 *Quantum computing with Qiskit*. <https://doi.org/10.48550/arXiv.2405.08810>

117 Khalid, A. (2024). *Stac*. <https://github.com/abdlahkhalids/stac>.

118 Leung, D., Vandersypen, L., Zhou, X., Sherwood, M., Yannoni, C., Kubinec, M., & Chuang,
119 I. (1999). Experimental realization of a two-bit phase damping quantum code. *Physical
120 Review A*, 60(3), 1924–1943. <https://doi.org/10.1103/physreva.60.1924>

121 PACE. (2017). *Partnership for an Advanced Computing Environment (PACE)*. <http://www.pace.gatech.edu>

122 Ryan-Anderson, C. (n.d.). *PECOS*. <https://github.com/PECOS-packages/PECOS>

123 Shor, P. W. (1994). Algorithms for quantum computation: Discrete logarithms and factoring.
124 *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 124–134.
125 <https://doi.org/10.1109/SFCS.1994.365700>

126 Shor, Peter W. (1995). Scheme for reducing decoherence in quantum computer memory. *Phys.
127 Rev. A*, 52, R2493–R2496. <https://doi.org/10.1103/PhysRevA.52.R2493>

128 tqec. (n.d.). *Tqec*. <https://github.com/tqec/tqec>

129 Viola, L., & Lloyd, S. (1998). Dynamical suppression of decoherence in two-state quantum
130 systems. *Phys. Rev. A*, 58, 2733–2744. <https://doi.org/10.1103/PhysRevA.58.2733>

131 Wille, R., Berent, L., Forster, T., Kunasaikaran, J., Mato, K., Peham, T., Quetschlich,
132 N., Rovara, D., Sander, A., Schmid, L., Schoenberger, D., Stade, Y., & Burgholzer,
133 L. (2024). The MQT handbook: A summary of design automation tools and software

¹³⁸ for quantum computing. *IEEE International Conference on Quantum Software (QSW)*.
¹³⁹ <https://doi.org/10.1109/QSW62656.2024.00013>

DRAFT