QECirc: A Community-Driven, FAIR Library of Quantum Error Correction Circuits

on behalf of the QECirc contributors

Ludwig Schmid[1], Tom Peham[1], and Robert Wille[1,2]
[1]Chair for Design Automation, Technical University of Munich, Munich, Germany
[2]MQSC, Garching near Munich, Germany
contribute@qecirc.com qecirc.com

***Abstract*—Quantum error correction (QEC) underpins fault-tolerant quantum computing, but the circuits that realize QEC codes (encoders, logical state-preparation and gate routines, and syndrome-extraction schedules) are scattered across papers and one-off repositories in incompatible formats, often documented too sparsely to reconstruct or reuse. We present *QECirc*, a community-driven, open library that makes QEC circuits Findable, Accessible, Interoperable, and Reusable (FAIR). Each entry links to its underlying code, ships in Stim and OpenQASM with interactive visualizations, and is checked for correctness before publication. In short, where a catalog such as the Error Correction Zoo describes the codes, QECirc provides the concrete circuits that realize them.**

I. Introduction and Motivation

Useful quantum computing is widely expected to require quantum error correction (QEC), in which logical qubits are redundantly encoded into entangled states of many noisy physical qubits [1]. Realizing this requires more than a good code: it requires concrete *circuits* that prepare encoded states, map data into and out of the code space, extract error syndromes, and apply logical gates. Preparing logical states is a central primitive for universal fault-tolerant computation, and its cost, especially its two-qubit-gate count and depth, contributes directly to the overhead of workflows such as magic-state distillation and cultivation [2].

Consequently, encoder and logical state-preparation synthesis is an active research area: reinforcement learning for hardware-adapted fault-tolerant circuits [3], greedy, rollout-based, and SMT-based encoder synthesis [2], and AI-optimized graph decimation [4]. These works report substantial reductions in gate count and depth for codes from the Steane and Golay codes [5] to the 144-qubit gross code [6], yet the resulting circuits are typically buried in paper appendices, ad-hoc scripts, or private repositories, in whatever format a particular group happens to use. As a result, researchers repeatedly re-derive the same circuits, and cross-work comparisons remain difficult. Code catalogs such as the Error Correction Zoo [1] have become standard references for QEC *codes*, but no analogous format-agnostic resource exists for the *circuits* that implement them.

QECirc fills this gap. It is a community-driven, open library of QEC circuits, browsable at qecirc.com, designed around the FAIR principles for scientific data [7], in which every published circuit is checked for correctness and every code carries its stabilizer check matrices for direct reproducibility.

II. Background and Related Work

Stim [8] is a de-facto standard for stabilizer circuits and their fast simulation, OpenQASM [9] is widely used for expressing quantum circuits, and tools such as Quirk and Crumble [10] help inspect small circuits interactively. The Error Correction Zoo [1] catalogs codes, and toolkits such as the Munich Quantum Toolkit (MQT) [11] construct and decode them.

The circuits themselves are the focus of an active research community. A wide range of methods synthesizes and optimizes encoders and logical state-preparation circuits [2], [3], [4], [12], [13], [14], [15], [16], fault-tolerant syndrome-extraction circuits and gadgets [17], [18], [19], [20], and logical gates from code automorphisms [21], building on classical and learned Clifford and linear-reversible (CNOT) synthesis routines [22], [23], [24], [25], [26], [27]. QECirc is *complementary*: rather than reproducing a taxonomy or synthesis algorithm, it collects, standardizes, and republishes the circuits such methods produce on one searchable platform.

III. The QECirc Library

Data model. QECirc links four entities: *codes*, *circuits*, the *tools* that generate them, and the *papers* that describe them. A code is identified by its parameters $[\![n, k, d]\!]$, descriptive tags, and its stabilizer and logical check matrices. Codes are canonicalized on entry, so different descriptions of the same code collapse to a single entry. Each circuit is attached to a code and a task: encoding, logical Pauli-eigenstate preparation, fault-tolerant flag gadgets, or logical Clifford gates. Every circuit records its provenance (the generating tool and originating paper), computed metrics such as qubit count, depth, and two-qubit gate count, plus connectivity and device tags, and reminds the user to cite the original work on copy or download.

Tools. A curated directory catalogs the software used to produce the circuits, ranging from reinforcement-learning circuit discovery [3], fault-tolerant state-preparation and encoding synthesis [12], [14], [2], and Clifford and CNOT synthesis engines [25], [26], [27] to flag gadgets [16], automorphism-based logical gates [21], and qLDPC toolkits [20]. Each

tool is linked to its repository, its papers, and the circuits it contributed, crediting the generating methods.

Interface. Circuits are presented grouped by code. A ranked search finds entries by code name, alias, tag, or source paper; codes can be filtered by parameters and tags, circuits by their metrics, sorted, or flagged as favorites. Circuit views preserve qubit coordinates and can overlay derived detectors and logical observables. Each circuit can be exported to Stim or OpenQASM and links to live Crumble and Quirk views for in-browser reuse.

IV. Verification and Reproducibility

Every submission is processed by an automated ingestion pipeline that extracts metrics, identifies the implemented code, and validates correctness before publication. It checks that encoders map the all-zero input into the code space and that state preparations satisfy every stabilizer and prepare the claimed logical basis state, for CSS and non-CSS codes alike, and derives detector and observable annotations for Stim-based simulation. Since each code exposes its check matrices, users can independently verify any result.

V. FAIR and Community

These properties make QECirc a FAIR [7] resource for QEC circuits. Entries are *findable* by code parameters, task, and tags, each with a stable identifier and machine-readable metadata for search engines and AI assistants; *accessible* through an open website and public GitHub repository, with no login or paywall; *interoperable* through standard formats (Stim and OpenQASM) that common simulators consume; and *reusable*: openly licensed, with provenance and tool attribution on every entry.

QECirc is moreover developed as open community infrastructure. Contributions are submitted through guided templates and validated automatically, and the platform is open source, so the community can extend both the collection and the site. The first external contributions [15] have already been merged and credited on the site. Such a shared resource reduces duplicated effort, enables fair comparison of synthesis methods against a common baseline, and lowers the barrier to obtaining a working circuit for a given code and task.

VI. Outlook: A Zoo for Circuits

Our longer-term aim is for QECirc to become, for QEC *circuits*, what the Error Correction Zoo [1] is for codes: a single, trusted, community-driven reference where well-engineered circuits are found, shared, and improved. Planned work includes broader task coverage (syndrome-extraction schedules [17]), larger qLDPC code families such as bivariate-bicycle codes [6], richer hardware-aware metadata, and deeper cross-linking with the Zoo. QECirc is in active development, already spanning hundreds of verified circuits across a wide range of codes and tasks, and we invite the wider quantum error correction community to browse, use, and, above all, contribute at qecirc.com.

Acknowledgment

We thank the QECirc contributors (everyone in the QEC community who has submitted, reviewed, or improved an entry), whose collective effort makes this library possible; the current list is maintained at qecirc.com. QECirc is supported by a Unitary Foundation micro-grant (https://unitary.foundation/grants/2026_qecirc) and by the Chair for Design Automation at the Technical University of Munich. The authors acknowledge funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program grant agreement No. 101001318 and No. 101114305, and the Munich Quantum Valley (MQV), which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus. Furthermore, this work was supported by the BMFTR under grant numbers 13N17298 and 01MQ25001I, the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant numbers 563402549 and 563436708. *GenAI:* an initial draft of this abstract was prepared with a generative AI system (Anthropic Claude) and then reworked and finalized by the authors, who take responsibility for all content.

References

[1] V. Albert and P. Faist, "The error correction zoo," [Online]. Available: https://errorcorrectionzoo.org, 2022.

[2] T. Peham, M. Steinberg, R. Wille, and S. Heußen, "Synthesis and optimization of encoding circuits for fault-tolerant quantum computation," arXiv:2605.15266, 2026.

[3] R. Zen *et al.*, "Quantum circuit discovery for fault-tolerant logical state preparation with reinforcement learning," *Phys. Rev. X*, vol. 15, p. 041012, 2025, arXiv:2402.17761.

[4] M. Doherty *et al.*, "Fast stabilizer state preparation via AI-optimized graph decimation," arXiv:2603.17743, 2026.

[5] A. Steane, "Error correcting codes in quantum theory," *Phys. Rev. Lett.*, vol. 77, no. 5, pp. 793–797, 1996, arXiv:quant-ph/9601029.

[6] S. Bravyi *et al.*, "High-threshold and low-overhead fault-tolerant quantum memory," *Nature*, vol. 627, pp. 778–782, 2024, arXiv:2308.07915.

[7] M. Wilkinson *et al.*, "The FAIR guiding principles for scientific data management and stewardship," *Sci. Data*, vol. 3, p. 160018, 2016.

[8] C. Gidney, "Stim: a fast stabilizer circuit simulator," *Quantum*, vol. 5, p. 497, 2021, arXiv:2103.02202.

[9] A. Cross *et al.*, "OpenQASM 3: A broader and deeper quantum assembly language," *ACM Trans. Quantum Comput.*, vol. 3, no. 3, pp. 1–50, 2022, arXiv:2104.14722.

[10] C. Gidney, "Quirk: a drag-and-drop quantum circuit simulator," [Online]. Available: https://algassert.com/quirk.

[11] R. Wille *et al.*, "The MQT handbook: A summary of design automation tools and software for quantum computing," in *Proc. IEEE Int. Conf. Quantum Software (QSW)*, 2024, arXiv:2405.17543.

[12] T. Peham *et al.*, "Automated synthesis of fault-tolerant state preparation circuits for quantum error correction codes," *PRX Quantum*, vol. 6, p. 020330, 2025, arXiv:2408.11894.

[13] N. Rengaswamy, R. Calderbank, S. Kadhe, and H. Pfister, "Logical Clifford synthesis for stabilizer codes," *IEEE Trans. Quantum Eng.*, vol. 1, p. 2501217, 2020, arXiv:1907.00310.

[14] L. Schmid *et al.*, "Deterministic fault-tolerant state preparation for near-term quantum error correction: Automatic synthesis using Boolean satisfiability," in *Proc. Design, Automation and Test in Europe (DATE)*, 2025, arXiv:2501.05527.

[15] L. Colmenarez *et al.*, "Unitary fault-tolerant encoding of Pauli states in surface codes," arXiv:2601.05113, 2026.

[16] D. Forlivesi and D. Amaro, "Flag at origin: a modular fault-tolerant preparation for CSS codes," arXiv:2508.14200, 2025.

[17] Y. Liu *et al.*, "AlphaSyndrome: Tackling the syndrome measurement circuit scheduling problem for QEC codes," arXiv:2601.12509, to appear in Proc. ASPLOS, 2026.

[18] R. Chao and B. Reichardt, "Quantum error correction with only two extra qubits," *Phys. Rev. Lett.*, vol. 121, p. 050502, 2018, arXiv:1705.02329.

[19] C. Chamberland and M. Beverland, "Flag fault-tolerant error correction with arbitrary distance codes," *Quantum*, vol. 2, p. 53, 2018, arXiv:1708.02246.

[20] M. Kang *et al.*, "QUITS: A modular QLDPC code circuit simulator," *Quantum*, vol. 9, p. 1931, 2025, arXiv:2504.02673.

[21] H. Sayginel *et al.*, "Fault-tolerant logical Clifford gates from code automorphisms," arXiv:2409.18175, 2024.

[22] J. Cossio *et al.*, "AlphaCNOT: Learning CNOT minimization with model-based planning," arXiv:2604.13812, 2026.

[23] S. Bravyi and D. Maslov, "Hadamard-free circuits expose the structure of the Clifford group," *IEEE Trans. Inf. Theory*, vol. 67, no. 7, pp. 4546–4563, 2021, arXiv:2003.09412.

[24] K. Patel, I. Markov, and J. Hayes, "Optimal synthesis of linear reversible circuits," *Quantum Inf. Comput.*, vol. 8, no. 3–4, pp. 282–294, 2008, arXiv:quant-ph/0302002.

[25] M. Webster, S. Koutsioumpas, and D. Browne, "Heuristic and optimal synthesis of CNOT and Clifford circuits," arXiv:2503.14660, 2025.

[26] T. Goubault de Brugière, S. Martiel, and C. Vuillot, "A graph-state based synthesis framework for Clifford isometries," *Quantum*, vol. 9, p. 1589, 2025, arXiv:2212.06928.

[27] T. Goubault de Brugière and S. Martiel, "Faster and shorter synthesis of Hamiltonian simulation circuits," arXiv:2404.03280, 2024.