Qsyn: A Developer-Friendly Quantum Circuit Synthesis Framework for NISQ Era and Beyond

Mu-Te Lau*†, Chin-Yi Cheng*†, Cheng-Hua Lu*‡, Chia-Hsu Chuang*†,
Yi-Hsiang Kuo†, Hsiang-Chun Yang†, Chien-Tung Kuo†, Hsin-Yu Chen†, Chen-Ying Tung†, Cheng-En Tsai†,
Guan-Hao Chen†, Leng-Kai Lin†, Ching-Huan Wang†, Tzu-Hsu Wang†, Chung-Yang Ric Huang†

†National Taiwan University, R.O.C., ‡Stanford University, U.S.
josh.lau2011@gmail.com, chin-yi.cheng@utexas.edu, cyhuang@ntu.edu.tw

***Abstract*—In this paper, we introduce Qsyn, a novel quantum circuit synthesis (QCS) framework designed to facilitate the research, development, testing, and experimentation of QCS algorithms and tools. Our framework is more developer-friendly than other modern QCS frameworks in three aspects: (1) Qsyn provides a comprehensive command-line interface that enables developers to design various testing scenarios with ease and conduct flexible experiments on their algorithms. This feature significantly streamlines the development process, making it more efficient and user-friendly. (2) Qsyn offers detailed access to multiple data representations at different abstraction levels of quantum circuits. This capability allows developers to optimize their algorithms extensively, gaining deeper insights and control over the structure and behavior of quantum circuits. By understanding the intricacies of circuit design, developers can achieve higher levels of optimization and performance in their algorithms. (3) Qsyn implements a rigorous development flow and environment to help developers maintain high-quality standards using modern software engineering practices, including robust quality assurance measures like regression testing, continuous integration and continuous delivery (CI/CD) pipelines, and code linting. We demonstrate Qsyn's superior performance through fair comparisons with PyZX [1], highlighting its efficiency and optimization capabilities. By providing a unified and user-friendly development environment, Qsyn empowers researchers and developers to prototype, implement, and evaluate their QCS algorithms effectively.**

***Index Terms*—Quantum Computing, Quantum Systems Software, Quantum Software Engineering**

I. Introduction

To push the advancement of the quantum circuit synthesis (QCS) algorithms in the future, we need a framework that offers a friendly environment for more QCS algorithm/tool developers to easily contribute their advanced ideas and conduct thorough experiments. As a result, we have open-sourced Qsyn, a developer-friendly QCS framework, to aid further development in this field. Our main contributions are:

1) Providing a unified and user-friendly developing environment so researchers and developers can efficiently prototype, implement, and evaluate their QCS algorithms with standardized tools, language, and data structures.
2) Assisting the developers with a robust and intuitive interface that can access low-level data directly to provide developers with unique insights into the behavior of their algorithms during runtime.
3) Enforcing robust quality-assurance practices, such as regression tests, continuous integration-and-continuous delivery (CI/CD) flows, linting, etc. These methodologies ensure that we provide reliable and efficient functionalities and that new features adhere to the same quality we strive for.

With Qsyn, developers can easily accelerate the implementation and evaluation of new QCS algorithms by leveraging the provided data structures and development environments.

II. Framework and Functionality of Qsyn

Qsyn embodies end-to-end synthesis by invoking functionalities for each synthesis stage as commands. It is designed with extensibility in mind, leveraging a data-oriented approach that focuses on robust data management and manipulation capabilities. Below, we introduce the core functionalities in Qsyn (https://github.com/DVLab-NTU/qsyn) for QCS.

1) High-level synthesis: Qsyn can process various specifications for quantum circuits by supporting syntheses with ROS Boolean oracle synthesis flow [2], [3] and Gray-code unitary matrix synthesis [4]–[6].

2) Gate-level synthesis: Qsyn can adapt to various optimization targets by offering different routines. For example, ZX-calculus-based- [1], [7] and tableau-based- [8]–[10] optimization routine aim to reduce T-count or 2-qubit-gate count.

For ZX-based methods, Qsyn has implementation advantages over PyZX. Qsyn revises the original algorithms in two aspects: (1) adopt a more compact ZX-diagram representation for MCT gates and an early-stopping strategy to avoid producing redundant graph complexity [11] (2) improves the conversion routine from ZX-diagrams to circuits. Specifically, the "gadget removal" strategy during conversion was improved to extract fewer CZ gates, contributing to a more compacted quantum circuit.

On the other hand, tableau, another data structure in Qsyn, represents Pauli rotation and phase polynomial which is used in QCS approaches. This representation unification eliminates

This work is supported by the National Science and Technology Council, R.O.C., Project No.: NSTC 113-2119-M-002-024.

979-8-3315-4137-8/24/$31.00 ©2024 IEEE
DOI 10.1109/QCE60285.2024.10392

frequent data conversions required when switching between these two synthesis paradigms.

3) Device mapping: Qsyn provides device mapping synthesis mainly based on [12]. Qsyn can also target a wide variety of quantum devices by addressing their available gate sets and topological constraints.

4) Verification and testing: Qsyn can directly convert small circuits (≤ 10 qubits) into a tensor and checks for equivalence numerically. As for larger circuits, symbolic verification can be applied by first appending the reversed copy of the circuit to its end. Then, we can invoke an optimization routine to reduce the composed circuit to identity.

III. Extending Qsyn

Qsyn offers a variety of capabilities that facilitate users' expanding of Qsyn for their own algorithms and applications. The extension is initiated with the design of the usage flow based on Qsyn's command-line interface (CLI), which processes user inputs and handles command execution. Then with the key capabilities described below, Qsyn provides a robust and adaptable framework for quantum circuit synthesis:

A. Self-defined gate type

Qsyn offers an extensible and highly expressive quantum circuit model. In addition to fundamental gate types, it allows developers to freely add new types of quantum gates. This capability enables developers and researchers to combine a wide range of quantum operations, simplifying the management of these complex tasks.

B. Shell interoperability

Developers can integrate Qsyn with external quantum synthesis tools, enhancing workflow flexibility through employing scripts. This shell interoperability becomes particularly useful when comparing optimization results between Qsyn and other synthesis tools.

C. Development of new algorithms

One of Qsyn's main missions is to assist developers in creating and assessing new synthesis algorithms. This is made possible by its powerful and easy-to-maintain argument parser, along with its useful CLI utilities.

IV. Comparision

We compare Qsyn's efficiency and optimization power with PyZX [1]. In Table I, we performed gate-level synthesis to examine the run time and memory usage of Qsyn against PyZX.

Compared to PyZX, Qsyn tended to optimize circuits with shorter program runtimes and lower memory usage. This distinction was particularly notable in circuits such as *gf2^128*, *hwb12*, and *urf4*, where PyZX encounters TLE and/or MLE (Time/Memory Limit Exceeded). For circuit statistics, Qsyn achieved either smaller or equal R_Z-counts and, on average, shallower circuits. This is attributed to Qsyn's integration of ZX-calculus and Tableau combined methods, further compressing rotation counts.

TABLE I
COMPARISON WITH PYZX [1] ON THE GATE-LEVEL SYNTHESIS STAGE.
TIME UNIT: s; MEMORY UNIT: MiB

Circuit	PyZX [1]				Qsyn			
name	$\#R_Z$	Depth	Time	Mem	$\#R_Z$	Depth	Time	Mem
hwb8	3517	15197	113.6	210.7	**3460**	15605	11.25	229.2
hwb9	43572	**176425**	5982	1142	**43565**	185294	516.7	6090
hwb10	15891	79892	2343	837.7	**15681**	79893	1808	4880
hwb12	MLE				**85611**	597419	4350	7471
sym9	**6450**	**31485**	373.0	270.2	**6450**	33078	25.45	1299
sym10	**11788**	60933	1247	429.7	**11788**	**60485**	114.4	892.1
urf2	5065	**18824**	350.7	308.4	**5007**	19267	6.45	175.6
urf4	MLE				**97036**	**389621**	9811	8709
urf6	33026	**129584**	10527	671.0	**31568**	133269	6751	15902
adder_433	**1536**	3802	43607	241.8	**1536**	4298	19.5	213.2
adder_64	**224**	670	41.57	243.8	**224**	683	0.3	23.59
gf2^128	TLE				**65664**	613170	1586	1802
gf2^64	16448	106078	23037	341.8	**16338**	67854	284.3	408.2
gf2^32	**4128**	18571	826.3	188.5	**4128**	11718	9.34	68.77
gf2^16	**1040**	5279	46.64	146.0	**1040**	3267	1.08	12.74
mult_350	TLE				TLE			
mult_45	**634**	**2702**	281.1	169.5	**634**	2987	31.05	184.1
qugan_395	**2360**	7092	46040	914.6	2361	3812	111.5	142.8
qugan_111	**657**	2326	568.9	215.9	**657**	1114	1.65	8.75

Acknowledgement

This work is supported by the National Science and Technology Council, R.O.C., Project No.: NSTC 113-2119-M-002-024.

References

[1] A. Kissinger and J. Van De Wetering, "PyZX: Large scale automated diagrammatic reasoning," in *Proc. Electronic Proceedings in Theoretical Computer Science (EPTCS)*, vol. 318, pp. 229–241, May 2020.

[2] G. Meuli, M. Soeken, M. Roetteler, and G. De Micheli, "ROS: Resource-constrained oracle synthesis for quantum computers," in *Proc. Electronic Proceedings in Theoretical Computer Science (EPTCS)*, vol. 318, pp. 119–130, May 2020.

[3] G. Meuli, M. Soeken, M. Roetteler, N. Bjorner, and G. D. Micheli, "Reversible pebbling game for quantum memory management," in *Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 288–291, Mar. 2019.

[4] A. M. Krol, A. Sarkar, I. Ashraf, Z. Al-Ars, and K. Bertels, "Efficient decomposition of unitary matrices in quantum circuit compilers," *Applied Sciences*, vol. 12, p. 759, Jan. 2022.

[5] M. Nakahara and T. Ohmi, *Quantum computing: from linear algebra to physical realizations*. CRC press, 2008.

[6] V. V. Shende, I. L. Markov, and S. S. Bullock, "Minimal universal two-qubit controlled-NOT-based circuits," *Physical Review A*, vol. 69, p. 062321, June 2004.

[7] A. Kissinger and J. Van De Wetering, "Reducing the number of non-Clifford gates in quantum circuits," *Physical Review A*, vol. 102, p. 022406, Aug. 2020.

[8] F. Zhang and J. Chen, "Optimizing T gates in Clifford+T circuit as $\pi/4$ rotations around Paulis," Mar. 2019.

[9] V. Vandaele, S. Martiel, S. Perdrix, and C. Vuillot, "Optimal hadamard gate count for Clifford+ *T* synthesis of Pauli rotations sequences," *ACM Transactions on Quantum Computing*, vol. 5, pp. 1–29, Mar. 2024.

[10] L. E. Heyfron and E. T. Campbell, "An efficient quantum compiler that reduces *T* count," *Quantum Science and Technology*, vol. 4, p. 015004, Sept. 2018.

[11] C.-H. Lu, "Dynamic quantum circuit optimization by ZX-calculus using Qsyn," July 2023.

[12] C.-Y. Cheng, C.-Y. Yang, Y.-H. Kuo, R.-C. Wang, H.-C. Cheng, and C.-Y. R. Huang, "Robust qubit mapping algorithm via double-source optimal routing on large quantum circuits," *ACM Transactions on Quantum Computing*, Aug 2024. Just Accepted.