

QContext: Context-Aware Decomposition for Quantum Gates

Ji Liu^{*§}, Max Bowman[†], Pranav Gokhale[‡], Siddharth Dangwal[¶],
Jeffrey Larson^{*}, Frederic T. Chong^{‡¶} and Paul D. Hovland^{*}

^{*} Mathematics and Computer Science Division, Argonne National Laboratory

[†] Department of Electrical & Computer Engineering, Rice University,

[‡] Super.tech, a division of Inflection,

[¶] Department of Computer Science, University of Chicago

Abstract—In this paper we propose QContext, a new compiler structure that incorporates context-aware and topology-aware decompositions. Because of circuit equivalence rules and resynthesis, variants of a gate-decomposition template may exist. QContext exploits the circuit information and the hardware topology to select the gate variant that increases circuit optimization opportunities. We study the basis-gate-level context-aware decomposition for Toffoli gates and the native-gate-level context-aware decomposition for CNOT gates. Our experiments show that QContext reduces the number of gates as compared with the state-of-the-art approach, Orchestrated Trios [12].

Index Terms—quantum computing, circuit synthesis, compiler optimization

I. INTRODUCTION

Quantum logic gates are the backbones of quantum information processing (QIP). The quantum logic gates in a program typically need to be decomposed to the basis gate set (ISA) in an assembly language. Since the target hardware may have limited connectivity, several recent works [10], [12] propose topology-aware decompositions that minimize the gate cost when mapping to a target hardware topology. However, these decomposition approaches use the same template to decompose the same type of quantum gates. Decomposing the gates with fixed templates lacks the opportunity to fully explore the circuit optimizations.

Gate context represents the predecessors and the successors in the directed acyclic graph (DAG) representation of the circuit. Since the different gate contexts could induce different circuit optimization opportunities, the quantum gate decomposition should be aware of the context. Here we use an example to demonstrate the effectiveness of context-aware gate decomposition. When the Toffoli gate in Figure 1b is decomposed in the canonical template with six CNOT gates, there is no gate cancellation and circuit resynthesis opportunity. However, since the Toffoli gate is a self-inverse gate, inverting all gates in the canonical template will result in another decomposition template. As shown in Figure 1b, we can perform gate cancellation and two-qubit block resynthesis optimizations when decomposing the Toffoli with the inversed template. Based on the circuit equivalence rules and resynthesis, several variants of a gate-decomposition template might exist. We develop our

context-aware decomposition approach QContext to identify the gate context structures and explore the circuit optimization opportunities.

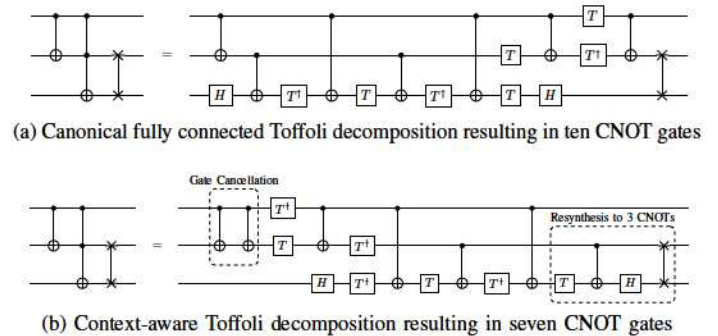


Fig. 1: Two Toffoli decompositions with different total CNOT gate counts

QContext provides both topology-aware and context-aware decomposition. QContext first finds the gate decomposition variants compatible with the target hardware topology. Then QContext selects the best gate decomposition variant based on successors' and predecessors' information in the DAG representation of the circuit. QContext also performs context-aware decomposition when decomposing the CNOT gates to the native cross-resonance (CR) gates and single-qubit gates. We propose new templates for the CNOT gate to CR gate decomposition. We implemented QContext in Qiskit-Terra and compared it with the state-of-the-art approach Orchestrated Trios. Our experiments show that by combining the basis-gate-level and the native-level context-aware decomposition, QContext reduces both two-qubit and single-qubit gate counts in quantum circuits.

II. BACKGROUND

Basis Gates and Native Gates: In order to run a quantum program on real hardware, the complex quantum gates need to be decomposed into the basis gates in an assembly language such as OpenQASM [8]. The basis gates in OpenQASM include single-qubit rotations and two-qubit CNOT gates.

However, the CNOT gates may not be natively supported by the quantum devices. The basis gates in the assembly language should be decomposed into a sequence of native

[§]Email: ji.liu@anl.gov

gates supported by the target hardware technology. In this paper we focus on the CR gate since it is the native gate for operating fixed-frequency superconducting devices such as IBM's superconducting systems. We expect context-aware decompositions to be applicable to other native gates such as CPHASE and iSWAP on Google [2] and Rigetti [5] devices.

Quantum Gate Decomposition: Optimal quantum gate decomposition is crucial for quantum compilation. Extensive research has been done to find efficient decompositions for unitary matrices. Besides the generalized decomposition, researchers have proposed structured templates [32], [33] to decompose certain types of quantum gates.

With the recent development in quantum hardware, several researchers have proposed topology-aware decompositions [6], [21] that minimize the gate cost when mapping to a target hardware topology. Toffoli gate is one of the most commonly used three-qubit gates in quantum information processing. Duckering et al. [12] proposed a compiler structure Orchestrated Trios, which efficiently routes and decomposes the Toffoli gates. They observed that it is more efficient to preserve the Toffoli gate during routing instead of routing each individual CNOT gate in the Toffoli gate decomposition. The approach first decomposes the quantum program to the one-, two-, and three-qubit gates and routes the gates. Then a second decomposition step is performed, which decomposes the Toffoli gate according to the hardware connectivity.

Quantum Optimization: The state-of-the-art quantum compilers mostly exploit circuit optimizations at the basis gate level. The Qiskit [1], $t|ket\rangle$ [34], and Cirq [7] compilers contain optimization passes that identify multiqubit blocks and resynthesize the blocks with KAK decomposition [23] to reduce circuit cost. Gate cancellation [24] is another useful optimization. Quantum gates may commute, and the compiler can reorder the gates to optimize the circuits and improve routing with gate cancellation [25]. Other optimizations include cross-talk mitigation [28], peephole optimization [29], and dynamical decoupling [30].

III. QCONTEXT

QContext performs basis-gate-level context-aware decomposition for the Toffoli gates and performs native-gate-level context-aware decomposition for the CNOT gates. Figure 2 shows the compilation flow of QContext. The boxes with the grey color backgrounds are the original compilation steps in Trios and Qiskit. The boxes with the orange color backgrounds are the compilation steps introduced and modified in QContext. Given an input quantum program, QContext first decomposes the gates to 1-qubit gates, 2-qubit gates, and Toffoli gates. Not decomposing the Toffoli gate allows the routing algorithm to capture the program structure and reduce the routing overhead [12]. The compiler performs qubit mapping and routing for the gates. We propose a gate library that contains the gate decomposition variants for the Toffoli and CNOT gates. The gate library contains 32 Toffoli gate variants and 14 CNOT gate variants. Each gate variant is associated with a *variant_tag*. The compiler specifies searches in the

library and returns the best matching gate variant based on the hardware topology and the gate context. Then, the compiler performs circuit optimizations. After the optimizations, the basis gates need to be decomposed into native gates. For each CNOT gate in the circuit, the compiler finds the CNOT gate variant that minimizes the number of single-qubit rotations after optimization.

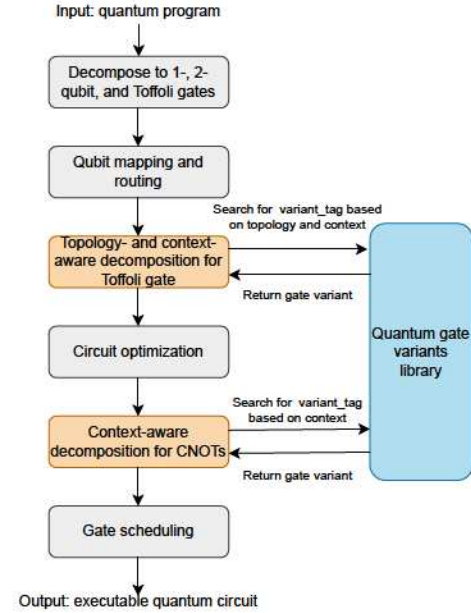


Fig. 2: Compilation flow of QContext. The steps introduced by QContext are marked with an orange background.

IV. BASIS-GATE-LEVEL DECOMPOSITION

We use the Toffoli gate as an example to discuss the format and the strategies for generating gate variants.

A. Variant_tag

We label the gate variants with different tags to differentiate them. The *variant_tag* for an n -qubit gate is a tuple with five elements:

$$\text{variant_tag} = (\text{pre_tag}, \text{suc_tag}, \text{design_tag}, \text{topo_tag}, \text{opt_tag}) \quad (1)$$

The first element *pre_tag* indicates the position of the CNOT gates at the beginning of the circuit that can be canceled or optimized with the predecessors. As shown in Figure 3, the first CNOT gate is between q_0 and q_2 . The *pre_tag* is set to "02" for this gate variant. The second element *suc_tag* indicates the position of the CNOTs at the end of the circuit that can be canceled or optimized with the successors. The last two CNOTs in the figure are between q_0 and q_1 , and the *suc_tag* is set to "10." The order of the bits indicates the control qubit and the target qubit.

The *design_tag* is an integer that indicates the basic design of the template. The gate variants that are generated by applying the equivalence rules to the canonical Toffoli decomposition will have the same *design_tag* = 0. However,

the gate variants based on resynthesis will have different designs and should have different *design_tag*. The *topo_tag* suggests the connectivity of the physical qubits. *topo_tag* can have four values: F, L0, L1, and L2. “F” or “L” indicates that the three qubits are fully connected or linearly connected. The integer represents the id of the qubit that is connected to the other two qubits. The last element *opt_tag* is used to differentiate the gate variants with similar structures but with different optimization opportunities. *opt_tag* can be either “O” or “I,” which differentiates the original decomposition and the inversed decomposition. The *variant_tag* provides an efficient way of defining and searching gate variants.

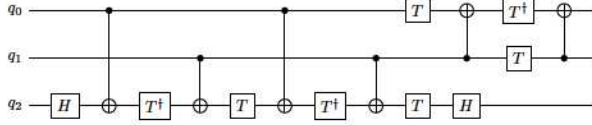


Fig. 3: Fully connected Toffoli variant by permuting control qubits, *variant_tag* = (02, 10, 0, F, O)

B. Toffoli Gate Variants

We can generate the gate variants based on the circuit equivalence rules [14] and resynthesis. We used three strategies described below to find gate variants.

1) Self-inverse: For the gates that are self-inverse, we can place the gates in the decomposition template in a reversed order and inverse each gate to get a new gate variant. Since the front layer of the circuit permutes with the last layer of the circuit, the *variant_tag* after inversion is specified by permuting *pre_tag* with *suc_tag*. The *opt_tag* changes from “O” to “I.” Other self-inverse gates include the CNOT gate, SWAP gate, Bridge gate, and Fredkin gate.

2) Permute control and target qubits: For a gate with multiple control qubits, we can permute the control qubits in a template to generate new gate variants. Figure 3 shows the gate variant that is generated based on the canonical decomposition and the permutation of control qubits q_0, q_1 . The *variant_tag* is specified by switching the associated qubits in the *pre_tag* and *suc_tag*. The *design_tag*, *topo_tag*, and *opt_tag* remain to be the same after permutation.

When the target gate is a NOT gate, we can also permute the control qubit with the target qubit by converting the multicontrolled NOT gate to the multicontrolled Z gate [14]. The multicontrolled Z gates are symmetrical; thus any qubit can be the target qubit.

3) Resynthesis: Based on equivalence rules, we can find many gate variants. However, some structures cannot be generated with the equivalence rules. One solution is to resynthesize [10], [35] the unitary matrix to the quantum circuit with the expected structures. A new *design_tag* will be assigned to the synthesized template.

We can apply the equivalence rules to the canonical 8-CNOT linear Toffoli decomposition to generate gate variants with linear connectivity. The 8-CNOT linear Toffoli decomposition is assigned with a *design_tag* equal to 1, since it is

different from the fully connected design. Figure 4 shows one of the linear Toffoli gate variants. Since q_2 is connected to q_0 and q_1 , the *topo_tag* is set to “L2.”

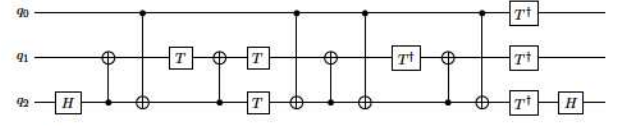


Fig. 4: Toffoli gate variant with linear connectivity, *variant_tag* = (21, 02, 1, L2, O)

C. Variant_tag Calculation

Here we describe the method to specify the *variant_tag* based on hardware topology and gate context. For each Toffoli gate node in the DAG, the compiler first determines the *topo_tag* based on the connectivity of the physical qubits. Then for each gate variant with the *topo_tag*, we can estimate the CNOT gate count reduction after optimization. Since the number of three-qubit gate variants is limited, the compiler performs an exhaustive search in the library and returns the gate variant with the maximum CNOT gate count reduction.

V. NATIVE-GATE-LEVEL DECOMPOSITION

A. Variant_tag

First, we introduce the format of the CNOT gate *variant_tag*. Two orientations of the CR gate are possible. The generalized CNOT gate decomposition circuit is a CR gate sandwiched by four U3 gates. The U3 gate is the generic single-qubit rotation gate with three angles θ, ϕ , and λ . Since the physical qubit connectivity for the CNOT gate is always linear, we do not need the *topo_tag* and *opt_tag*. The CNOT gate *variant_tag* is a tuple with twelve elements that specify the four U3 gates in the template and an *ori_tag* to specify the orientation of the CR gate.

$$\text{variant_tag} = (\theta_1, \phi_1, \lambda_1, \dots, \theta_4, \phi_4, \lambda_4, \text{ori_tag})$$

The *variant_tag* allows us to quickly find the inverse gate and calculate the native gate count. $U3(-\theta, -\phi, -\lambda)$ is the inverse gate of $U3(\theta, \phi, \lambda)$. The generalized U3 gate requires two $R_x(90)$ pulses and three R_z gates. The R_z gates [26] are implemented in software by frame change and do not introduce any noise. When decomposing the CNOT gate, the compiler first specifies the *ori_tag* based on the orientation of the target physical qubit connection. Then, for each *variant_tag* that contains the correct *ori_tag*, the compiler estimates the total number of $R_x(90)$ pulses after optimization and selects the one with the fewest $R_x(90)$ gates.

B. CNOT Gate Variants

We have three strategies for finding CNOT gate variants. **1) Self-inverse:** The CNOT gate is a self-inverse gate. We can place the native gates in a reversed order and invert each native gate to get a new gate decomposition variant.

2) Polarity switch: The CR native gate is implemented with a positive half-CR and a negative half-CR gate to mitigate

noise [36]. We can instead implement the CR gate as a negative half-CR followed by a positive half-CR gate. This polarity switch [16], [17] introduces a side effect that appends two $R_x(180)$ gates to the left and the right of the CR native gate. We can also combine the polarity switch and the gate inverse.

3) Resynthesis: We used a numerical optimization approach to find novel CNOT gate decompositions. We may treat the discovery of novel decomposition of the CNOT unitary matrix as a nonlinear least-squares optimization problem. An objective function can be defined as the square of the Frobenius norm of the difference between the parameter-generated unitary matrix and the target unitary matrix, where the U3 parameters are adjusted to minimize the objective function. Several standard least-squares optimization algorithms such as Levenberg–Marquardt [27] and Broyden–Fletcher–Goldfarb–Shanno [20] can be used.

We found gate variants that have more gate optimization opportunities. For instance, the ladder-shaped circuit structure shown in Figure 5 widely exists in quantum circuits [22]. Here we use a numerical approach to find the gate variant that has X gate on the top left and another X gate on the bottom right. When there is a sequence of CNOTs, the X gates in between will cancel out. In Figure 5, the single-qubit gates that can be optimized are marked with a dashed box.

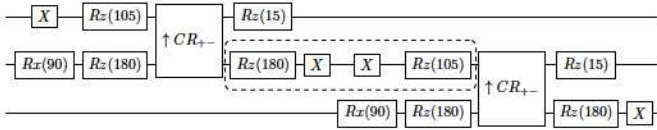


Fig. 5: Ladder-shaped CNOT structure and gate decomposition

VI. METHODOLOGY

Implementation: We implement our QContext framework on the open-source quantum computing framework Qiskit [1]. The version of Qiskit-Terra is 0.18.3, and our implementation is publicly available at <https://github.com/revilooliver/context-aware-decomposition>. We compare our results with the Orchestrated Trios compiler implemented in Qiskit.

Benchmarks: The benchmarks in our experiments are derived from the Toffoli benchmarks in Orchestrated Trios [11] and the reversible circuits [39]. Besides the benchmarks with Toffoli gates, we include the VQE UCCSD ansatz for LiH [31] and QAOA for Max-Cut [13].

VII. EVALUATION AND DISCUSSION

In this section we show the single- and two-qubit gate numbers using QContext compared with Trios on `ibmq_montreal` which has heavy-hex topology.

The CR gate and single-qubit gate reductions are shown in Table I. All the Toffoli gates are decomposed into the linearly connected gate variants with 8 CNOTs. Since our optimization focuses on the Toffoli gates and the CNOT gates, to distinguish the effect of qubit routing and our approach, we separate the CR gates for qubit routing and the CR gates for

the benchmark. CR_r represents the CR gates that form the SWAP gate and are not involved in the optimization. A better routing algorithm may reduce the CR_r number, but that is beyond the scope of the current discussion. CR_b represents the number of CR gates that are needed for implementing the benchmark. ΔCR_b is the percentage change in the number of CR_b : $\Delta CR_b = 1 - CR_b(QContext) / CR_b(Trios)$. $\Delta \#sx$ is the percentage change in the number of total sx rotations: $\Delta \#sx = 1 - \#sx(QContext) / \#sx(Trios)$. In the native gates, only the 90-degree x -axis rotation sx and the 180-degree x -axis rotation x gate will introduce noise. Here we calculate the total number of x -axis 90-degree rotations as $\#sx$. An x gate is counted as two sx gates. The CNOT gate context-aware decomposition also optimizes the inserted SWAP gates, so we do not differentiate the routing circuit when calculating $\#sx$.

TABLE I: Number of CR gates of QContext in comparison with Trios [12] on `ibmq_montreal`

Circuit	Trios			QContext			Comparison		
	#Q	CR_r	CR_b	#sx	CR_b	#sx	ΔCR_b	$\Delta \#sx$	T_c/T_t
cnx_half [15]	5	0	35	112	29	64	17.1%	42.9%	1.14
sym6 [39]	10	99	128	631	100	402	21.9%	21.7%	1.09
cnx_dirty [3]	11	102	146	666	119	520	18.5%	21.9%	1.02
cnx_half [15]	19	132	289	1204	230	841	20.4%	30.1%	1.26
cnx_log [4]	19	195	163	972	129	799	20.9%	17.8%	1.32
c_adder [9]	20	507	196	1828	184	1579	6.1%	13.6%	1.19
t_adder [37]	20	483	220	1859	191	1532	13.2%	17.6%	1.04
incrementer [15]	5	105	461	1585	421	1114	8.7%	29.7%	1.02
grover [18]	9	201	724	2524	609	1848	15.9%	26.8%	1.16
QAOA [13]	10	168	96	672	96	592	—	11.9%	1.04
UCCSD [31]	8	654	274	2348	274	2022	—	13.9%	1.10
Geo mean							14.7%	20.9%	1.12

#Q denotes the number of qubits. `t_adder` is the `takahashi_adder` [37]. `c_adder` is the `cuccaro_adder` [9].

T_c/T_t is the ratio between total transpilation time of QContext and Trios.

As shown in Table I, QContext reduces the number of single-qubit rotations $\#sx$ for all the benchmarks and reduces CR_b for all the benchmarks that contain Toffoli gates. The geometric mean of ΔCR_b for the benchmarks that contain Toffoli gates is 14.7%. The geometric mean of $\Delta \#sx$ is 20.9%. The results for the QAOA benchmark and the UCCSD benchmark show the effectiveness of CNOT gate context-aware decomposition. Note that QAOA and the UCCSD benchmark do not contain any Toffoli gate, therefore, they will not have any CNOT gate reduction. We also compare the compilation time. The choice of using a library of pre-synthesized templates makes QContext more scalable in terms of compilation time.

VIII. CONCLUSION

In this paper we propose QContext, a new compiler structure that incorporates context-aware and topology-aware decompositions. We highlight that quantum gate decomposition should be aware of the context. QContext reduces both the single- and two-qubit gate counts by being aware of the basis-gate-level and native-gate-level gate context.

IX. ACKNOWLEDGEMENT

This material is based upon work supported by Q-NEXT, one of the U.S. Department of Energy Office of Science (DOE-SC) National Quantum Information Science Research Centers and the Office of Advanced Scientific Computing Research,

Accelerated Research for Quantum Computing program. This work is funded in part by EPIQC, an NSF Expedition in Computing, under award CCF-1730449; in part by STAQ under award NSF Phy-1818914; in part by the US Department of Energy Office of Advanced Scientific Computing Research, Accelerated Research for Quantum Computing Program; and in part by the NSF Quantum Leap Challenge Institute for Hybrid Quantum Architectures and Networks (NSF Award 2016136) and in part based upon work supported by the U.S. Department of Energy, Office of Science, National Quantum Information Science Research Centers. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. FTC is Chief Scientist for Quantum Software at Infleqion and an advisor to Quantum Circuits, Inc.

REFERENCES

- [1] ANIS, M., et al., Qiskit: An Open-source Framework for Quantum Computing. (2021)
- [2] F. Arute, et al., "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.
- [3] J. M. Baker, C. Duckering, A. Hoover, and F. T. Chong, "Decomposing quantum generalized Toffoli with an arbitrary number of ancilla," *arXiv:1904.01671*, 2019.
- [4] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Physical Review A*, vol. 52, no. 5, p. 3457, 1995.
- [5] S. A. Caldwell, et al., "Parametrically activated entangling gates using transmon qubits," *Physical Review Applied*, vol. 10, no. 3, p. 034050, 2018.
- [6] X. Cheng, Z. Guan, and W. Ding, "Mapping from multiple-control Toffoli circuits to linear nearest neighbor quantum circuits," *Quantum Information Processing*, vol. 17, no. 7, pp. 1–26, 2018.
- [7] Cirq developers, "Cirq," 2021. [Online]. Available: <https://zenodo.org/record/5182845>
- [8] A. W. Cross, A. Javadi-Abhari, T. Alexander, N. de Beaudrap, L. S. Bishop, S. Heidel, C. A. Ryan, J. Smolin, J. M. Gambetta, and B. R. Johnson, "OpenQASM 3: A broader and deeper quantum assembly language," *arXiv:2104.14722*, 2021.
- [9] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, "A new quantum ripple-carry addition circuit," *arXiv quant-ph/0410184*, 2004.
- [10] M. G. Davis, E. Smith, A. Tudor, K. Sen, I. Siddiqi, and C. Iancu, "Towards optimal topology aware quantum circuit synthesis," in *International Conference on Quantum Computing and Engineering*. IEEE, 2020, pp. 223–234.
- [11] C. Duckering, J. M. Baker, P. Gokhale, and A. Litteken, "Quantum circuit benchmarks," 2020. [Online]. Available: <https://github.com/jmbaker94/quantumcircuitbenchmarks>
- [12] C. Duckering, J. M. Baker, A. Litteken, and F. T. Chong, "Orchestrated Trios: compiling for efficient communication in quantum programs with 3-qubit gates," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2021, pp. 375–385.
- [13] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv:1411.4028*, 2014.
- [14] J. C. Garcia-Escartin and P. Chamorro-Posada, "Equivalent quantum circuits," *arXiv:1110.2998*, 2011.
- [15] C. Gidney, "Constructing large controlled NOTs," 2015. [Online]. Available: <https://algassert.com/circuits/2015/06/05/Constructing-Large-Controlled-Not.html>
- [16] P. Gokhale, T. Tomesh, M. Suchara, and F. T. Chong, "Faster and more reliable quantum SWAPs via native gates," *arXiv:2109.13199*, 2021.
- [17] M. A. Bowman, P. Gokhale, J. Larson, J. Liu, and M. Suchara, "Hardware-conscious optimization of the quantum Toffoli gate," *arXiv:2209.02669*, 2022.
- [18] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, 1996, pp. 212–219.
- [19] K. Gui, T. Tomesh, P. Gokhale, Y. Shi, F. T. Chong, M. Martonosi, and M. Suchara, "Term grouping and travelling salesperson for digital quantum simulation," *arXiv:2001.05983*, 2020.
- [20] J. D. Head and M. C. Zerner, "A Broyden–Fletcher–Goldfarb–Shanno optimization procedure for molecular geometries," *Chemical Physics Letters*, vol. 122, no. 3, pp. 264–270, 1985.
- [21] S. Hu, D. Maslov, M. Pistoia, and J. Gambetta, "Efficient circuits for quantum search over 2D square lattice architecture," in *Proceedings of the 56th Annual Design Automation Conference*, 2019, pp. 1–2.
- [22] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, pp. 242–246, 2017.
- [23] B. Kraus and J. Cirac, "Optimal creation of entanglement using a two-qubit gate," *Physical Review A*, vol. 63, no. 6, p. 062309, 2001.
- [24] D. Maslov, G. W. Dueck, D. M. Miller, and C. Negrevergne, "Quantum circuit simplification and level compaction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 3, pp. 436–444, 2008.
- [25] J. Liu, P. Li, and H. Zhou, "Not all swaps have the same cost: A case for optimization-aware qubit routing," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2022, pp. 709–725.
- [26] D. C. McKay, C. J. Wood, S. Sheldon, J. M. Chow, and J. M. Gambetta, "Efficient Z gates for quantum computing," *Physical Review A*, vol. 96, no. 2, p. 022330, 2017.
- [27] J. J. Moré, "The Levenberg–Marquardt algorithm: Implementation and theory," in *Numerical Analysis*. Springer, 1978, pp. 105–116.
- [28] P. Murali, D. C. McKay, M. Martonosi, and A. Javadi-Abhari, "Software mitigation of crosstalk on noisy intermediate-scale quantum computers," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 1001–1016.
- [29] J. Liu, L. Bello, and H. Zhou, "Relaxed peephole optimization: A novel compiler optimization for quantum circuits," in *2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. IEEE, 2021, pp. 301–314.
- [30] K. N. Smith, G. S. Ravi, P. Murali, J. M. Baker, N. Earnest, A. Javadi-Abhari, and F. T. Chong, "Timestitch: Exploiting slack to mitigate decoherence in quantum circuits," *ACM Transactions on Quantum Computing*, vol. 4, no. 1, pp. 1–27, 2022.
- [31] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature Communications*, vol. 5, no. 1, pp. 1–7, 2014.
- [32] C. Jones, "Low-overhead constructions for the fault-tolerant Toffoli gate," *Physical Review A*, vol. 87, no. 2, p. 022328, 2013.
- [33] R. Cleve and J. Watrous, "Fast parallel circuits for the quantum Fourier transform," in *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*. IEEE, 2000, pp. 526–536.
- [34] S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington, and R. Duncan, "[t]ket: A retargetable compiler for NISQ devices," *Quantum Science and Technology*, vol. 6, no. 1, p. 014003, 2020.
- [35] E. Smith, M. G. Davis, J. M. Larson, E. Younis, C. Iancu, and W. Lavrijsen, "LEAP: Scaling numerical optimization based synthesis using an incremental approach," *ACM Transactions on Quantum Computing*, doi:10.1145/3548693.
- [36] N. Sundaresan, I. Lauer, E. Pritchett, E. Magesan, P. Jurcevic, and J. M. Gambetta, "Reducing unitary and spectator errors in cross resonance with optimized rotary echoes," *PRX Quantum*, vol. 1, no. 2, p. 020318, 2020.
- [37] Y. Takahashi, S. Tani, and N. Kunihiro, "Quantum addition circuits and unbounded fan-out," *arXiv:0910.2530*, 2009.
- [38] R. Wille and D. Große, "Fast exact Toffoli network synthesis of reversible logic," in *IEEE/ACM International Conference on Computer-Aided Design*. IEEE, 2007, pp. 60–64.
- [39] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler, "RevLib: An online resource for reversible functions and reversible circuits," in *38th International Symposium on Multiple Valued Logic*, 2008, pp. 220–225.