THE UNIVERSITY OF CHICAGO

DESIGN, OPTIMIZATION, AND SIMULATION OF SCALABLE QUANTUM COMPUTING SYSTEMS

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY
XIN-CHUAN WU

CHICAGO, ILLINOIS
MARCH 2021

TABLE OF CONTENTS

LIST OF FIGURES vi

LIST OF TABLES ix

ACKNOWLEDGMENTS x

ABSTRACT xi

1 INTRODUCTION 1

2 THESIS STATEMENT 3

3 ARTISAN: SCALABLE QUANTUM CIRCUIT OPTIMIZATION USING HIERARCHICAL SYNTHESIS 4
- 3.1 Introduction 4
- 3.2 Background 7
  - 3.2.1 Principles of Quantum Computation 8
  - 3.2.2 Quantum Circuit Synthesis 8
- 3.3 Scalable Circuit Optimization using Synthesis 9
  - 3.3.1 Artisan Overview 10
  - 3.3.2 Physical Qubit Mapping 11
  - 3.3.3 Circuit Partitioning 11
  - 3.3.4 Quantum Circuit Synthesis 15
  - 3.3.5 Circuit Composition 16
- 3.4 Experimental Setup 16
  - 3.4.1 Benchmarks 17
  - 3.4.2 Experimental Parameters 18
- 3.5 Evaluation 18
  - 3.5.1 CNOT Reduction 20
  - 3.5.2 Impact of Synthesis Distance 20
  - 3.5.3 Running on Real Hardware 22
  - 3.5.4 Running on Noise Simulation 22
  - 3.5.5 Scalability 24
  - 3.5.6 Discussion 26
- 3.6 Related Work 27
- 3.7 Conclusion 29

4 TILT: ACHIEVING HIGHER FIDELITY ON A TRAPPED-ION LINEAR-TAPE QUANTUM COMPUTING ARCHITECTURE 30
- 4.1 Background 35
  - 4.1.1 Principles of Quantum Computation 35
  - 4.1.2 Trapped-Ion QC Systems 35
- 4.2 TILT Architecture 37

4.2.1 Feasibility . . . 38
4.2.2 Gate Selection . . . 38
4.2.3 Compared with QCCD . . . 39
4.2.4 TILT Challenges . . . 40
4.3 LinQ for TILT Architecture . . . 41
4.3.1 LinQ Overview . . . 41
4.3.2 Native Gate Decomposition . . . 43
4.3.3 Qubit Mapping and Swap Insertion . . . 43
4.3.4 Tape Movement Scheduling . . . 46
4.3.5 TILT Simulation . . . 48
4.4 Experimental Setup . . . 49
4.4.1 Benchmarks . . . 49
4.4.2 Simulation Parameters . . . 50
4.5 Evaluation . . . 51
4.5.1 LinQ Swap Insertion Performance . . . 51
4.5.2 Architecture Comparison . . . 54
4.5.3 Compilation Time and Execution Time . . . 55
4.6 Trapped-Ion Scaling . . . 57
4.7 Related Work . . . 58
4.8 Conclusion . . . 60

5 FULL-STATE QUANTUM CIRCUIT SIMULATION BY USING DATA COMPRESSION . . . 61
5.1 Background and Related Work . . . 64
5.1.1 Principles of Quantum Computation . . . 64
5.1.2 Quantum Circuit Simulation . . . 65
5.1.3 Data Compression Techniques . . . 67
5.2 Simulation Design . . . 69
5.2.1 Overview of Our Simulation Flow . . . 70
5.2.2 MCDRAM Memory Configuration . . . 71
5.2.3 Integration Details . . . 72
5.2.4 Compressed Block Cache . . . 73
5.2.5 Simulation Checkpoint . . . 74
5.2.6 MPI Configuration . . . 74
5.2.7 Variable Error Bound Compression . . . 74
5.2.8 Lower Bounds on Simulation Accuracy . . . 75
5.3 Adaptive Compression Optimized for Quantum Circuit Simulation . . . 76
5.3.1 Assessment of Existing Error-Bounded Lossy Compressors . . . 77
5.3.2 Optimizing the Compression Strategy . . . 79
5.4 Evaluation . . . 85
5.4.1 Experimental Setup . . . 85
5.4.2 Scalability . . . 86
5.4.3 Benchmarks . . . 86
5.4.4 Experimental Results . . . 88
5.4.5 Discussion . . . 89

5.5 Conclusion 90

6 FUTURE WORK 95
6.1 Quantum Circuit Optimization using Synthesis 95
6.1.1 Approximate Synthesis 95
6.1.2 Pulse-Level Optimization 95
6.1.3 Artisan in Parallel 96
6.2 TILT Architecture Scaling 96
6.2.1 Sympathetic Cooling 96
6.2.2 QCCD Architectures 97
6.2.3 Photonic Interconnects 97
6.3 Quantum Circuit Simulation by using Lossy Compression 98
6.3.1 Noise Simulation 98

7 CONCLUSION 99

A CURRICULUM VITAE 101

REFERENCES 105

LIST OF FIGURES

3.1 An example of Artisan circuit optimization. (a) The original 5-qubit circuit. (b)The original circuit is partitioned into three blocks, and each block only contains gates on 3 qubits. The first circuit block is only on q_0, q_1, and q_4. The second block is on q_1, q_2, and q_4. The third block is on q_0, q_1, and q_3. All gates after partitioning still respect the gate dependency in the original circuit. (c) The synthesized circuit. After running quantum synthesis for each block, the CNOT counts in the first and second block are reduced, and the third block still has 2 CNOTs. (d) The single-qubit gates can be combined to produce the final optimized circuit. This circuit is effectively equivalent to the original circuit but with fewer CNOT gates. 5

3.2 Our compilation framework for scalable circuit optimization using synthesis (Artisan). 6

3.3 Example of valid and invalid qubit groups. 13

3.4 An example of gate dependency graph and executable gates on a qubit group. . 14

3.5 Experimental platform used in our evaluation. 18

3.6 The numbers of CNOTs are reduced in the Artisan-optimized circuits. 19

3.7 Average CNOT count in a block. 19

3.8 Compilation time. 21

3.9 State infidelity due to the synthesis distance. Compared with the gate error on NISQ devices, the state infidelity due to synthesis distance is negligible. 21

3.10 Running optimized small circuits on IBM's Athens, a 5-qubit device. (Lower d_{TV} is better.) 23

3.11 Correlation between CNOT reduction rate and d_{TV} improvement measured on IBM's Athens. Each data point is a benchmark from the small set. The correlation coefficient is 78%. The results show that our CNOT reductions are indicative of fidelity savings. 23

3.12 Running optimized small circuits on Qiskit noise simulation. (Lower d_{TV} is better.) 24

3.13 Correlation between d_{TV} results on the real device and noise simulation. Each data point is a benchmark from the small set. The correlation coefficient is 98%. 24

3.14 Correlation between CNOT reduction rate and d_{TV} improvement measured on noise simulation. Each data point is a benchmark from the medium set. The correlation coefficient is 76%. 24

3.15 Realistic noise simulation results of d_{TV} under different levels of gate error. (Lower d_{TV} is better.) 25

3.16 The number of CNOTs in the optimized large-scale circuits. 26

3.17 Compilation time of large-scale circuit optimization. 26

3.18 CNOT reduction rate of large-scale circuits. 26

3.19 Circuit success rate under different levels of gate errors. (Higher circuit success rate is better.) 27

4.1 A trapped-ion linear-tape quantum computing architecture. Acousto-optic modulators (AOMs) target laser beams for quantum operations, which can be applied to ions in the execution zone. In order to perform gate operations on the other qubits, the entire ion chain is translated until the target qubit is moved into the execution zone. 32
4.2 (a) A two-qubit gate is applied to Q_1 and Q_2 (marked by blue). Since the distance between Q_1 and Q_2 is longer than the execution zone, a swap between Q_1 and Q_3 (marked by green) is needed. After the swap, a tape movement brings Q_1 and Q_2 in the execution zone for the two-qubit gate execution. (b) Suppose a two-qubit gate is applied to $\{Q_1, Q_2\}$ (marked by red) and another two-qubit gate is applied to $\{Q_3, Q_4\}$ (marked by blue). Two regular swaps are needed to make the two-qubit gates executable. (c) An opposing swap combines two regular swaps in opposite directions, and this swap makes both two-qubit gates executable. Thus, creating opposing swaps can reduce the total number of swaps. 33
4.3 Compare TILT with QCCD. g1, g2, g3, and g4 are two-qubit gates on four pairs of qubits (marked by blue, green, orange, and red). (a) For QCCD, when a qubit communicates with another qubit in the different trap, QCCD requires the following operations: i) Swap the qubit to the end of the current trap. ii) Split from the current chain. iii) Shuttle to the target trap. iv) Merge to the target chain. v) Interact with the target qubit. To execute the gates g1, g2, g3, and g4, QCCD needs to repeat the above operations four times. (b) For TILT, performing a tape movement would allow the operations of the desired two-qubit gates. . . 39
4.4 LinQ overview. Taking a quantum program and device specification as input, LinQ generates the executable codes, including a sequence of scheduled gates and tape movements, and then runs the simulation to output the success rate and run-time. 42
4.5 Considering two-qubit gates g_1 and g_2 on a system with the tape head size of L. Since d_{g_1} is $L-1$, only one tape head position is allowed for g_1 to be executed. As for g_2, there are three valid tape head positions because d_{g_2} is $L-3$. 44
4.6 Comparing LinQ swap insertion with the baseline swap insertion. 52
4.7 Success rates under different $MaxSwapLen$ restriction. For larger swap length, the circuit may need more tape moves. If the swap length is too short, however, the required number of tape moves increases as the total number of gates is increased. There is a swap length sweet spot depending on applications. For BV, the swap counts and move counts are the same so the lines are overlapped. . . . 53
4.8 Success rates on different device configurations (higher is better). 55

5.1 Example of two-qubit quantum state with single-qubit gate operations 69
5.2 Simulation overview 69
5.3 Amplitude index segments 72
5.4 Compressed block cache line 73
5.5 Normalized execution time for running 35-qubit random circuit simulations. . . 74
5.6 Minimum bounds for fidelity with an increasing number of gates at different error levels 77
5.7 Compression ratio of SZ vs. ZFP (absolute error) 78

5.8 Compression ratio of SZ,FPZIP,ZFP (relative error) 79
5.9 Illustration of Value changes of quantum circuit simulation data: the data exhibit a high spikiness such that existing lossy compressors cannot work effectively. . . 79
5.10 Compression ratio of 4 solutions (relative error) 83
5.11 Compress/decompression rates (relative error) 91
5.12 Distribution of maximum relative errors on compression of qaoa_36 and sup_36 (Solution C and D overlap) 92
5.13 Illustrating why Solution C/D leads to lower compression errors than Solution A/B 93
5.14 Distribution of normalized compression errors (Solution C) 93
5.15 Normalized execution time for running various sizes of simulations on a single node. 94
5.16 Scaling behavior of applying Hadamard gates on a 51-qubit system on Theta. . 94

LIST OF TABLES

3.1 Notations used in our algorithm. 12
3.2 List of benchmarks. 17

4.1 Notations used in this paper. 41
4.2 List of benchmarks. 49
4.3 LinQ compilation results. 56

5.1 Examples of supercomputers, their total memory capacity, and the maximum number of qubits they can simulate for arbitrary circuits. 61
5.2 Experimental results. The first row shows the memory requirements of the simulations without our techniques. The ratios of our system memory sizes to the required memory sizes are presented in the fourth row. The fifth row provides the simulation time and the breakdown. The minimum compression ratio during the simulation is shown in the last row. 87

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor Frederic T. Chong for his great support, kindness, and advice over the past four and half years. Thanks also to the rest of my dissertation committee, Hank Hoffmann and Yuri Alexeev, for their time and feedback.

I also thank my collaborators, Yongshan Ding, Dripto Debroy, Ken Brown, Sheng Di, and Franck Cappello, for their feedback and support. I am also grateful for my summer internships with Hal Finkel, Yuri Alexeev, and Costin Iancu that greatly expanded my expertise in quantum computing. I also appreciate the social and intellectual support from my labmates: Yunong, Yongshan, Adam, Reza, Pranav, Jonathan, Casey, Andrew, Sophia, Rich, Kate, and Gokul. I am thankful for the EPiQC team. I have learned a lot from all the EPiQC members.

In the end, I would like to thank my mother, Mei-Chi Cheng, and my wife, Linda Lan, for their great supports.

This work is funded in part by EPiQC, an NSF Expedition in Computing, under grants CCF-1730449; in part by STAQ under grant NSF Phy-1818914; in part by DOE grants DE-SC0020289 and DE-SC0020331; and in part by NSF OMA-2016136 and the Q-NEXT DOE NQI Center.

ABSTRACT

Quantum computing (QC) aims to solve certain computational problems beyond the capabilities of even the largest classical high-performance computers. By leveraging the quantum mechanical principles of superposition and entanglement, QC algorithms have the potential to revolutionize areas such as machine learning, quantum chemistry, and cryptography. To perform quantum advantages, a scalable quantum computing system is essential.

This thesis demonstrates techniques that address problems of the development of scalable quantum computers. The first part of the thesis describes our technique to perform scalable quantum circuit optimization by using synthesis. Traditional quantum synthesis can only be used for circuits on a handful of qubits. The proposed technique partitions the circuit into multiple small pieces of circuits and performs quantum synthesis individually. This technique enables quantum circuit synthesis for large circuits. The second section is the design and evaluation of the trapped-ion linear-tape architecture. This study provides an alternative idea towards scalable quantum computing systems. The last section of the thesis is our technique to increase the capability of quantum circuit simulation on classical supercomputing systems. Our hybrid data compression technique is introduced into the simulation of quantum computing. This work allows us to further push the limit of classical simulation, and hence verify a larger size of quantum computing systems.

CHAPTER 1

INTRODUCTION

Quantum computing (QC) aims to solve certain computational problems beyond the capabilities of even the largest classical high-performance computers. By leveraging the quantum mechanical principles of superposition and entanglement, QC algorithms have the potential to revolutionize areas such as machine learning [23], quantum chemistry [143, 98], and cryptography [156]. Recently, IBM, Rigetti, and Google demonstrated their QC devices up to 72 superconducting transmon qubits [91, 73, 147], while IonQ and Honeywell have recently made significant steps with trapped-ion QC devices [144, 93]. Current QC machines in the Noisy Intermediate-Scale Quantum (NISQ) era [145] are too small for large benchmarks and unable to support quantum error correction (QEC) [18, 76]. We can, however, run on the order of hundreds of quantum operations using on the order of hundreds of quantum bits (qubits).

The power of quantum computation grows exponentially with the number of qubits. Achieving a scalable quantum computing system is essential for performing quantum advantages. Therefore, we need to consider the simulation, design, and optimization of scalalbe quantum computing systems. This thesis covers three research components: increasing the capability of quantum circuit simulation, a detailed design and analysis of a novel scalable trapped-ion quantum computing architecture, and a quantum circuit optimization method for scalable quantum circuits.

The first part of the thesis describes our technique of scalable quantum circuit optimization by using quantum synthesis. Traditional quantum synthesis can only be used for circuits on a handful of qubits. Our technique partitions the circuit into multiple small pieces of circuits and performs quantum synthesis individually. This work enables quantum synthesis for large circuits. The second section is the design and evaluation of the trapped-ion linear-tape architecture. The scheduling algorithms are proposed to enable the realization of this system architecture. This work provides an alternative viable path towards scalable quan-

tum computing systems. The last section of the thesis describes our technique to increase the capability of quantum circuit simulation on classical supercomputing systems. Our data compression technique is introduced into the simulation of quantum computing. This work allows us to further push the limit of classical simulation, and hence verify a larger size of quantum computing systems.

CHAPTER 2

THESIS STATEMENT

This thesis demonstrates techniques that address the problems of architecting scalable quantum computing systems. Specifically, there are three techniques proposed to scale quantum circuit optimization, architecture design, and circuit simulation. The first technique is to enable scalable quantum circuit optimization by using divide and conquer and synthesis. The second work designs and evaluates techniques that achieve scalable trapped-ion quantum computing architecture. The third work applies data compression to full-state quantum circuit simulation to scale the simulation capability. Altogether, this thesis provides a viable path towards scalable quantum computing systems.

CHAPTER 3

ARTISAN: SCALABLE QUANTUM CIRCUIT OPTIMIZATION USING HIERARCHICAL SYNTHESIS

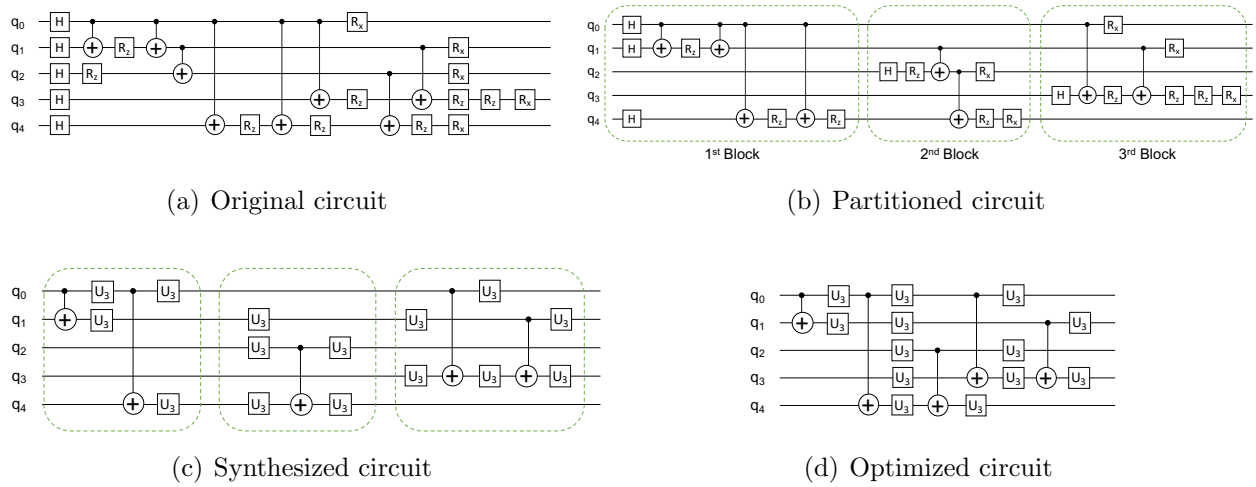
3.1 Introduction

Quantum Computing (QC) is expected to solve certain computational problems that even the largest classical supercomputers cannot solve. QC algorithms might have significant impact on areas such as quantum chemistry [143, 98], cryptography [156], and machine learning [23]. Recently, QC devices up to 72 quantum bits (qubits) have been demonstrated by IBM and Google [91, 73]. The current phase of QC is in the Noisy Intermediate-Scale Quantum (NISQ) era [145]. On NISQ devices, quantum gates are noisy and their count must be minimized to produce low-error circuits. In particular, two-qubit gates such as CNOTs are much noisier than single-qubit gates. For NISQ devices, shorter depth translates directly into higher circuit fidelity.

Quantum circuit synthesis provides an orthogonal circuit optimization method, which generates a circuit from its high level mathematical description such as unitary matrix. There are several existing studies in this field [153, 52, 59, 135, 3, 54, 51, 7, 104, 124, 68, 168, 182, 27]. Given an arbitrary quantum circuit, we can compute the corresponding unitary, and then use synthesis to generate a new circuit, which has a different sequence of gates from the original circuit, but will be equivalent in terms of unitary operator performed.

However, synthesis algorithms face an "exponential" scalability challenge. For a n-qubit circuit, the unitary size is $2^n \times 2^n$. The solving time of synthesis scales exponentially with the number of qubits [3, 59, 51]. Thus, synthesis is intractable for circuits beyond a handful of qubits.

To perform quantum synthesis for optimizing large scale circuits, we propose our hierarchical, block-by-block, optimization framework, called Artisan. Artisan uses a combination of partitioning and synthesis: 1) partition the circuit into independent sub-blocks whose size



(a) Original circuit

(b) Partitioned circuit

(c) Synthesized circuit

(d) Optimized circuit

Figure 3.1: An example of Artisan circuit optimization. (a) The original 5-qubit circuit. (b)The original circuit is partitioned into three blocks, and each block only contains gates on 3 qubits. The first circuit block is only on q_0, q_1, and q_4. The second block is on q_1, q_2, and q_4. The third block is on q_0, q_1, and q_3. All gates after partitioning still respect the gate dependency in the original circuit. (c) The synthesized circuit. After running quantum synthesis for each block, the CNOT counts in the first and second block are reduced, and the third block still has 2 CNOTs. (d) The single-qubit gates can be combined to produce the final optimized circuit. This circuit is effectively equivalent to the original circuit but with fewer CNOT gates.

can be successfully handled by synthesis; 2) re-generate each block using synthesis; and 3) re-assemble the circuit. Figure 3.1 shows an example of Artisan circuit optimization that partitions a 5-qubit circuit into 3-qubit blocks. In general, partitioning a n-qubit circuit into multiple k-qubit blocks ($k < n$) leads to replacing an algorithm with $O(exp(n))$ complexity with a sequence of calls to $O(exp(k))$ algorithms. The time-to-solution grows linearly with the number of k-qubit blocks, which is a useful tradeoff when compared to the exponential scaling with the number of qubits. After all blocks are synthesized, Artisan then puts the blocks back together to produce the final optimized circuit.

Artisan allows us to compile a quantum circuit to an optimized executable circuit for a target hardware. Artisan is a topology-aware compilation framework. The overall flow is described in Figure 3.2. Artisan includes four core modules: physical qubit mapping, circuit partitioning, quantum synthesis, and circuit composition. The input to Artisan is a quantum

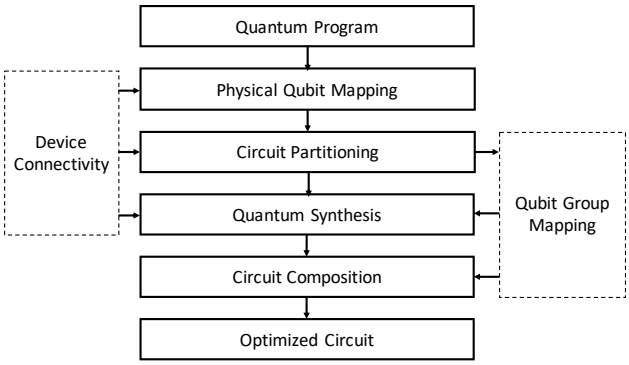


Figure 3.2: Our compilation framework for scalable circuit optimization using synthesis (Artisan).

circuit in a high-level quantum program, together with a description of the topology of the target processor. The first step in Artisan is topology mapping of the circuit to the target hardware. We leverage third party mappers and call into "traditional" compilers to perform mapping. This approach allows us to select the best mapper available for a given platform. We use the $t|ket\rangle$ compiler for physical qubit mapping as it is reported to effectively produce short circuits for several applications [160, 47]. We then run our partitioning algorithm to select tunable size blocks of the circuit that are independent of each other. In the synthesis procedure, each block is converted from its circuit format into a unitary matrix and re-synthesized with a synthesis tool. In this work, we use the state-of-the-art synthesis tool proposed in [51]. The final output of Artisan is the optimized circuit by stitching all the optimized blocks together.

For evaluation we use a series of NISQ circuits as our benchmarks. To understand the impact of the partitioning strategy we conduct experiments with 3- and 4-qubit blocks. We evaluate the quality of the generated circuits on IBM's Athens [92] device, as well as using noise simulation for medium-size circuits and on our analytical model for large-size circuits.

The main contributions of this work are:

- Our technique allows an exponential cost optimization tool to scale to large circuits, and is the first demonstration of successfully employing and scaling synthesis in the compilation toolchain for large circuit optimization.

- Artisan reduces the CNOT gate count well beyond the ability of existing compilers. Using a set of NISQ benchmarks, we show that Artisan can reduce the number of CNOT gates by 29.9% on average and up to 50% compared with circuits optimized by $t|ket\rangle$ compiler. These translate into direct fidelity improvements when running on the IBM's Athens device [48].

- We evaluate fidelity improvements using 3 metrics for 3 scaling regimes: small-size circuits using fidelity on real devices, medium-size circuits using fidelity using simulations with noise, and large-size circuits using CNOT reduction measured statically by our compiler. To validate the trends shown by our simulations, we show that there is a 98% correlation between our real-device results and our simulation results on small circuits. To show that our CNOT reductions are also indicative of fidelity savings, we show there is a 78% correlation to real devices on small circuits and 76% correlation to simulation on medium circuits.

- We present the sensitivity analysis by running Qiskit noise simulations to show the circuit fidelity improvements under different levels of gate errors. The results indicate that Artisan is important for NISQ devices.

- We demonstrate the scalability of our technique to optimize the circuits of 60+ qubits. This demonstration suggests that Artisan provides a viable path towards higher fidelity for circuits on large scale qubits.

3.2 Background

In this section, we briefly introduce the principles of quantum computation. We then present the relevant background on quantum circuit synthesis.

3.2.1 Principles of Quantum Computation

A qubit is a two-level quantum system, represented by two orthonormal computational basis states $|0\rangle$ and $|1\rangle$. The quantum state of a qubit can be described by any linear combination of $|0\rangle$ and $|1\rangle$: $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, where α and β are complex numbers satisfying $|\alpha|^2 + |\beta|^2 = 1$. More generally, the state of a n-qubit quantum system can be represented by using 2^n amplitudes.

A quantum circuit consists of a sequence of quantum gates on qubits, and a quantum gate is a unitary operator, U. The effect of a gate on a quantum state can be expressed as $|\psi'\rangle = U\,|\psi\rangle$. Gates are applied to one or many qubits simultaneously. Two-qubit controlled gates and arbitrary single-qubit gates are known to be universal [61]. CNOT gates and single-qubit rotation gates (R_x, R_y, R_z) constitute a commonly used universal gate set for quantum programming.

3.2.2 Quantum Circuit Synthesis

Quantum circuit synthesis is the process of taking a description of a desired unitary matrix and decomposing it into a sequence of smaller unitaries, representing the gates in a circuit that implements the target unitary. One of the earliest techniques was the Solovay-Kitaev algorithm[52, 135, 3], which combined a recursive matrix decomposition technique with numerical methods required for the base case. This method served as a proof-of-concept for the field of synthesis, but the circuits it produces are very long. One of the main goals of synthesis is to produce "short" circuits, minimizing metrics such as CNOT count, total gate count, and critical path length. CNOT count is of particular importance on NISQ devices, since they contribute significantly more to the overall noise and runtime of circuits than single-qubit gates. Because of this, CNOT count has been a particular target for synthesis approaches. The approach shown in [7] is able to find minimal length circuits according to a weighting of different gates, and demonstrates a significant advantage over the Solovay-Kitaev approach. The approaches shown in [104] and [59] focus on synthesis runtime, which

is another metric needed for synthesis to be practical. All of these techniques mentioned so far work solely with discrete gatesets, where only a finite number of base gates are allowed. However, most current quantum devices can perform continuously varying gates. For example, superconducting qubits use virtual Z rotations, which can be performed with any angle, and trapped ion qubits can perform X and Y rotations with any angle. These continuous gates can be used to perform any single qubit rotation, usually parameterized as $Z(\theta_1)X(\frac{\pi}{2})Z(\theta_2)X(\frac{\pi}{2})Z(\theta_3)$ or $X(\theta_1)Y(\theta_2)Z(\theta_3)$. Leveraging this capability, the KAK decomposition can perform optimal 2-qubit synthesis to perform any 2 qubit unitary with at most 3 CNOTs [168]. The approach shown in [153] uses the cosine-sine decomposition to produce universal gates for any number of qubits, which can perform any unitary simply by changing the parameters. Other approaches use numerical optimization of parameterized gates to find high quality circuits [124, 51, 182].

We employ the search-based numerical synthesis method described in [51]. This technique builds a search tree of CNOT structures filled in with parameterized single-qubit gates, and employs numerical optimization to evaluate the CNOT structure at each node in the search. A* search is used to find the shortest CNOT structure that can implement the desired unitary, and numerical optimization is used to find parameters for the single-qubit gates to produce a fully instantiated circuit. We choose this technique because of its ability to minimize CNOT count.

3.3 Scalable Circuit Optimization using Synthesis

Quantum synthesis is a powerful tool for circuit optimization. However, it is not scalable for large circuits. To optimize asymptotically large circuits, we develop our block-by-block optimization technique, called Artisan. Artisan allows us to compile a high-level quantum program to a hardware topology-aware executable circuit. Artisan takes the device qubit connectivity and a quantum program as inputs, and produces a compiled executable circuit, minimizing the CNOT gate count by using synthesis.

3.3.1 Artisan Overview

Figure 3.2 shows an overview of Artisan. It contains four core compiler components: physical qubit mapping, circuit partitioning, quantum synthesis, and the circuit composition procedure. The inputs for Artisan consist of a high-level quantum program and a target device connectivity description. First, Artisan performs physical qubit mapping to assign the logical qubits to physical qubits and to resolve all unexecutable two-qubit gates by adding swap gates. Since our goal is to produce an optimized topology-aware circuit, if we perform physical qubit mapping after quantum synthesis, there will be swap gates inevitably added into the final circuit. Therefore, we perform physical qubit mapping before quantum synthesis, so that the swap gates added into the circuit become a part of the input circuit for the circuit synthesis, and hence the final circuit would have a reduced CNOT gate count. Second, Artisan partitions the circuit into multiple small circuit blocks. Each block only interacts with a few qubits. The qubit group mapping is generated to indicate what qubits are involved for each circuit block. This qubit group mapping information will be used for the later circuit composition procedure. Since the number of qubits in a block reflects the size of the corresponding unitary matrix, we demonstrate the partitioning of 3-qubit and 4-qubit blocks in this work, but the block size can be further increased. In general, a larger unitary matrix would take exponentially longer time to decompose into a sequence of gates. A circuit partitioned with a larger block size would have fewer blocks, and each block tends to allow the synthesizer to decide circuit elements rather than the mapping algorithm, and hence the final optimized circuit would have fewer CNOTs. Thus, the block size is a trade-off between the time-to-solution and the quality of solution (CNOT reduction). Next, quantum synthesis is applied to the partitioned circuit blocks. We use the state-of-the-art synthesis tool proposed in [51] for our work. Finally, Artisan performs the circuit composition by following the qubit group mapping to concatenate the synthesized blocks to produce the final optimized circuit.

3.3.2 Physical Qubit Mapping

Given an input quantum circuit and the qubit connectivity graph, physical qubit mapping is to find an initial qubit mapping and insert swap gates to satisfy all two-qubit interactions and try to minimize the total number of swap gates and circuit depth in the final hardware-compliant circuit. Physical qubit mapping problem is NP-Complete [157]. Several approaches for solving this problem have been proposed [11, 175, 121, 48, 126, 40, 139, 110, 94, 160, 179, 187].

We design the physical qubit mapping as the first step of Artisan because we want to have the additional swaps gates to be part of the input circuit for the remaining optimization process, so that the optimization by quantum synthesis can reduce the CNOT gate count in the final circuit. In addition, we also find that most quantum applications have certain repeated CNOT patterns, and existing qubit mapping algorithms will perform better by leveraging the structure of these patterns. If the synthesis process runs before physical qubit mapping, the pattern will be destroyed, and the number of additional swap gates will increase in some cases. Thus, performing physical qubit mapping before synthesis can benefit the overall optimization.

We compare industrial compilers such as IBM Qiskit [48] and $t|ket\rangle$ compilers [160]. Since the $t|ket\rangle$ compiler produces shorter circuits in our experiments and is reported to produce shorter circuits for several quantum applications compared with other techniques [47], we adopt the $t|ket\rangle$ compiler for physical qubit mapping in our Artisan.

3.3.3 Circuit Partitioning

Artisan partitions a circuit into multiple small circuit blocks. Each block contains gates only on a small group of qubits. Figure 3.1 shows an example of partitioning. Each circuit block consists of 3 qubits (Figure 3.1(b)). Since the CNOT reduction would be higher when there are more CNOTs in a block, the goal of partitioning is to *find a block with the number of qubits and biggest number of CNOTs*. Thus, we propose our algorithm: we

Table 3.1: Notations used in our algorithm.

Notation	Definition
n	The total number of qubits in a circuit
k	The total number of qubits in a partitioned block
g_i	A gate operation. i is the index of the gate.
q_i	A qubit. i is the index of the qubit.
Q_i	A qubit group. i is the index of the group.
E_{Q_i}	A set of executable gates on Q_i
G	Gate dependency graph
B	A list of partitioned blocks
M	A list of qubit group mapping

use a greedy-based heuristic approach. This is an efficient approach; compared with other partitioning algorithms such as dynamic programming that would be limited by circuit depth, our heuristic algorithm is scalable for large-scale circuit partitioning. We summarize the notations used in our algorithm in Table 3.1.

Before our heuristic algorithm, we define a few terms used in our algorithm.

Qubit group. A qubit group is a set of qubits in a circuit block. For a n-qubit circuit, there are $\binom{n}{k}$ different combinations of qubit groups for a k-qubit block. For example, the circuit shown in Figure 3.1(a) is a 5-qubit circuit. For partitioning the circuit into 3-qubit blocks, $\{q_0, q_1, q_2\}, \{q_0, q_1, q_3\}, ...,$ and $\{q_2, q_3, q_4\}$ are possible qubit groups for a block. Figure 3.1(b) shows the circuit after partitioning. The qubit group of the 1st block is $\{q_0, q_1, q_4\}$, and the 2nd block is $\{q_1, q_2, q_4\}$, and the 3rd block is $\{q_0, q_1, q_3\}$. We use Q_i to denote a qubit group in this paper, where $i = 1, 2, ..., \binom{n}{k}$. The qubit group size k is limited to a small number due to the limitation of quantum synthesis. In this paper, we focus our analysis on the size of 3 and 4 qubits in a qubit group.

Valid qubit group. Since the input circuit for our partitioning algorithm is a physical qubit mapped circuit, all two-qubit gates are applied on the neighbor qubits. We can define *valid qubit groups* by considering device connectivity to reduce the search space in our algorithm. We map the qubit group onto the device connectivity graph. If a qubit group is a connected component, it is a valid qubit group; otherwise, it is an invalid group. Figure 3.3

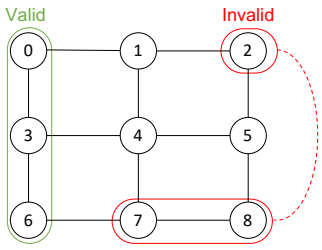
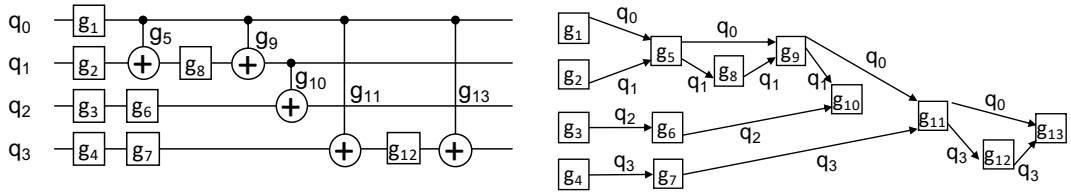


Figure 3.3: Example of valid and invalid qubit groups.

shows an example of a valid qubit group. Considering a 3×3 grid connectivity, the qubit group $\{q_0, q_3, q_6\}$ is a valid qubit group. However, the qubit group $\{q_2, q_7, q_8\}$ is an invalid qubit group. When partitioning a circuit, we only need to consider valid qubit groups.

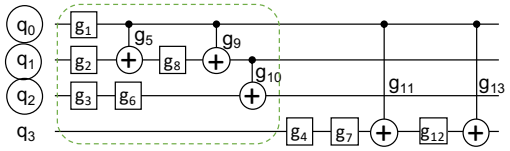
Gate dependency graph. We use a Directed Acyclic Graph (DAG) to represent the gate dependency of a circuit. In a DAG, a node represents a gate, and an edge is the qubit involved for the gate. A gate can be executed only when all the previous gates have been executed. Figure 3.4 shows an example of gate dependency graph. The DAG in Figure 3.4(b) is generated from the original circuit (Figure 3.4(a)). For example, the gate g_5 depends on g_1 and g_2 because q_0 is used for g_1 and g_5, and q_1 is used for g_2 and g_5. Thus, g_5 can be executed only after both g_1 and g_2 are executed.

Executable gates on a qubit group. A single-qubit gate on q_i can count as an executable gate on a qubit group Q_i only when $q_i \in Q_i$ and all previous gates on q_i are executable gates on Q_i. A two-qubit gate on (q_i, q_j) can count as an executable gate on a qubit group Q_i only when both $q_i, q_j \in Q_i$ and all previous gates on q_i and q_j are executable gates on Q_i. In Figure 3.4, for example, when we count the executable gates on the qubit group $\{q_0, q_1, q_2\}$, the gate g_9 is an executable gate because q_0 and q_1 are in the qubit group and all the previous gates on q_0 and q_1 are also executable gates. However, g_4 is not an executable gate on this qubit group because q_3 is not in the target qubit group. We use E_{Q_i} to denote the largest set of executable gates on the qubit group Q_i. We can get each E_{Q_i} by traversing the gate dependency graph. Figure 3.4(c) and Figure 3.4(d) show $E_{\{q_0,q_1,q_2\}}$ and $E_{\{q_0,q_1,q_3\}}$, respectively.

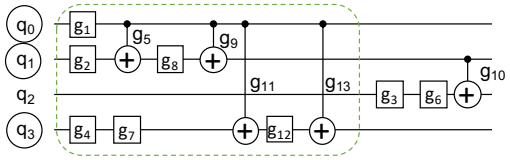


(a) Original circuit

(b) Gate dependency graph



(c) Executable gates on qubit group $\{q_0, q_1, q_2\}$.



(d) Executable gates on qubit group $\{q_0, q_1, q_3\}$.

Figure 3.4: An example of gate dependency graph and executable gates on a qubit group.

Algorithm 1 shows our partitioning procedure. First, we prepare the gate dependency graph G for the target circuit. Next, we collect the set of executable gates, E_{Q_i}, for each qubit group, and also compute the score for each E_{Q_i}. Since the objective is to find the biggest circuit block, we select the E_{Q_m} with the maximal score as the partitioned block, and save the Q_m as the qubit group mapping for this block. Once a circuit block is partitioned, we remove the block from the gate dependency graph G, and repeat the procedure to find the next circuit block partition until the gate dependency graph G is empty, which means the entire circuit is partitioned. Finally, a sequence of circuit blocks B and a list of qubit group mapping M are the output of the circuit partitioning process.

The heuristic cost function indicates the number of executable gates for a qubit group.

Algorithm 1 Circuit Partitioning

Convert the circuit into a gate dependency graph G.
$B \leftarrow \phi$
$M \leftarrow \phi$
while G is not empty **do**
    for each valid qubit group Q_i **do**
        $E_{Q_i} \leftarrow$ {Executable Gates on Q_i}
        Compute $Score(E_{Q_i})$
    end for
    Select the E_{Q_m} with the maximal score;
    $B.append(E_{Q_m})$
    $M.append(Q_m)$
    $G \leftarrow G - E_{Q_m}$
end while
Output the sequence of circuit blocks B and the list of qubit group mapping M

The general form of our heuristic cost function for a qubit group Q_i is shown as follows:

$$Score(E_{Q_i}) = N_{E_{Q_i}}, \tag{3.1}$$

where $N_{E_{Q_i}}$ is the number of executable CNOT gates on Q_i. Since the objective of partitioning is to find a block for quantum synthesis to minimize the CNOT count, a block with more CNOT gates can potentially achieve higher CNOT reduction. Thus, we use CNOT count as the score function.

The time complexity of our heuristic algorithm for partitioning a circuit into k-qubit blocks is $O(n^k g)$, where n is the number of qubits in the original circuit, g is the total number of gates, k is the number of qubits in a partitioned block.

3.3.4 Quantum Circuit Synthesis

After the circuit is partitioned into multiple blocks, Artisan applies quantum synthesis for each circuit block. We integrate the state-of-the-art synthesis tool proposed in [51] into our Artisan framework. For each circuit block, Artisan computes the corresponding unitary matrix, and this matrix is the input for the synthesis process. As the synthesized circuit should

respect the hardware topology, the qubit group for each block and the device connectivity are also the input parameters for the synthesis tool. With our block-based synthesis scheme, the time-to-solution for a n-qubit circuit is reduced from $O(exp(n))$ to $O(exp(k))$.

After the synthesis, there is an undesired synthesis distance due to numerical approximation, leading the final unitary at a distance from the original unitary [136, 51, 182]. As a result, the final unitary distance is in the range of 10^{-10} to 10^{-15}. In the NISQ era, when running a quantum circuit on a real machine, the unitary executed is different from the original intended unitary due to the presence of noise. Since gate errors are multiple orders of magnitudes larger than synthesis distances, these synthesis distances are insignificant. In Section 3.5.2 we will show that this distance only causes negligible impact on the overall state fidelity.

3.3.5 Circuit Composition

Once all circuit blocks have been synthesized, Artisan composes the entire circuit by stitching all circuit blocks. The list of qubit group mappings provide the qubit mapping information to connect blocks correctly. In general, the number of CNOTs in the synthesized block is less than in the original block. If the synthesized circuit block has an equal or greater number of CNOTs compared to the original block, Artisan will choose to use the original circuit block to the circuit composition to avoid unnecessary synthesis distance due to floating point errors. Once all blocks are put together, Artisan combines the adjacent single-qubit gates to further reduce the gate count, and produces the final optimized circuit.

3.4 Experimental Setup

In this section, we describe a set of benchmarks used in our evaluation as well as the experimental parameters.

3.4.1 Benchmarks

Table 3.2: List of benchmarks.

Size	Application	Qubits	Description
Small	QAOA5	5	QAOA on MaxCut problem
	TFIM5	5	Transverse-field Ising model
	MUL5	5	Multiplier arithmetic function
	ADDER4	4	Adder arithmetic function
	QFT5	5	Quantum Fourier transform
	HLF5	5	Hidden linear function
Medium	QAOA10	10	QAOA on MaxCut problem
	TFIM10	10	Transverse-field Ising model
	MUL10	10	Multiplier arithmetic function
	ADDER9	9	Adder arithmetic function
	QFT9	9	Quantum Fourier transform
	HLF9	9	Hidden linear function
Large	MUL60	60	Multiplier arithmetic function
	ADDER63	63	Adder arithmetic function
	QFT64	64	Quantum Fourier transform

To evaluate Artisan against real applications, we select multiple important quantum applications as our benchmarks. Table 3.2 lists the applications and brief description in our evaluation. We have three sets of applications according to the different sizes of the circuits. The first set consists of small-size circuits on 4 and 5 qubits; the second set is the medium-size of 9- and 10-qubit circuits; and the third set of benchmarks, large-size circuits, contains circuits beyond 60 qubits. In our evaluation, we run the small-size circuits on a 5-qubit quantum device, Athens, provided by IBM quantum experience platform to demonstrate how much circuit fidelity is improved through our optimization on a real NISQ machine. Next, we run our medium-size benchmarks by using Qiskit noisy simulators to show the fidelity improvement under different levels of gate errors. Finally, we use the large-size benchmarks to demonstrate the scalability of our Artisan technique.

Quantum Approximate Optimization Algorithm (QAOA) is a hybrid quantum-classical variational algorithm and it is one of the most important quantum algorithms in the NISQ era [65]. In our study, the QAOA application is a hardware-efficiency ansatz [129] for Max-Cut problem. Transverse Field Ising Model (TFIM) is for problems that study the time

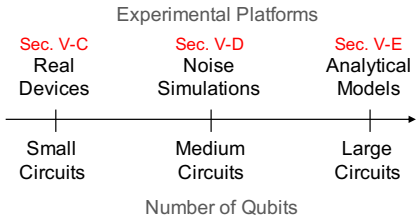


Figure 3.5: Experimental platform used in our evaluation.

evolution of chemical systems [155, 85]. Multiplier (MUL) [67] and adder [67, 49] benchmarks are important arithmetic functions in several quantum applications. Quantum Fourier transform (QFT) [95] is a kernel function in many quantum algorithms such as phase estimation algorithm [44], Shor's algorithm [156], and the algorithm for hidden subgroup problem [97]. Hidden Linear Function (HLF) is a quantum circuit solving a problem from a previous study [33].

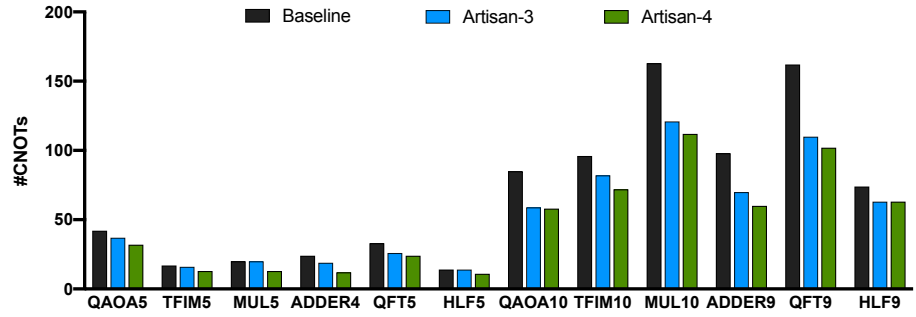
3.4.2 Experimental Parameters

We evaluate our Artisan with 3-qubit and 4-qubit blocks on different sizes of circuits. All Artisan processes and noisy simulations are carried out on a Ubuntu 16.04 system with Intel Xeon Silver 4110 32-core CPU (2.1 GHz) and 128GB RAM.

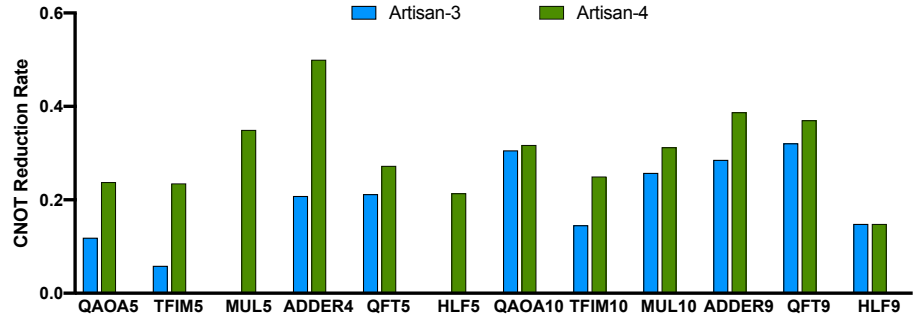
Figure 3.5 summarizes our fidelity analysis. For small circuits, we show the experimental results on IBM's Athens device (Section 3.5.3). For medium circuits, we run the experiments on noise simulations (Section 3.5.4). Finally, we use our analytical model to perform the fidelity results for large circuits (Section 3.5.5). The $t|ket\rangle$ compiler with the highest optimization level is the baseline in our evaluation. The $t|ket\rangle$ compiler is reported to effectively produce shorter circuits for several applications compared with other compilers [160, 47].

3.5 Evaluation

In our evaluation, we first show the CNOT reduction in the Artisan-optimized circuits compared to the baseline ($t|ket\rangle$) optimization, and we compare the compilation time using



(a) Total number of CNOTs in the circuits optimized by the baseline compiler and Artisan.



(b) CNOT reduction rate compared with the baseline compiler.

Figure 3.6: The numbers of CNOTs are reduced in the Artisan-optimized circuits.

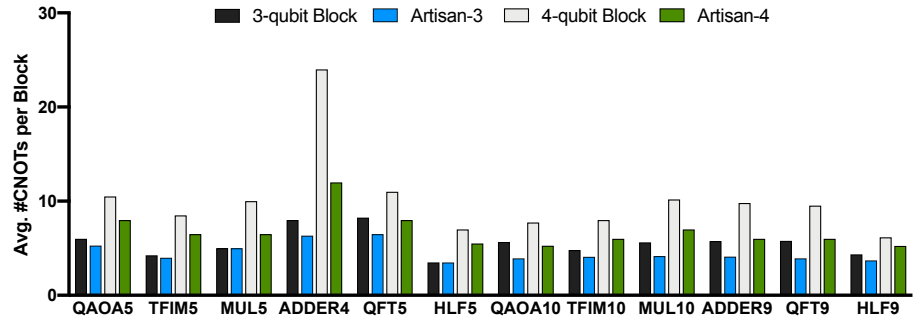


Figure 3.7: Average CNOT count in a block.

3-qubit and 4-qubit blocks. We then discuss the impact of synthesis distance on the quantum state. Next, we present the results running on a real quantum device, IBM's 5-qubit device (Athens), to prove that our technique is critical for the current devices. Then a sensitivity analysis is performed to show how much circuit fidelity is improved under different gate errors. We also demonstrate the scalability of Artisan by optimizing large circuits.

3.5.1 CNOT Reduction

We use the $t|ket\rangle$ compiler to map 4- and 5-qubit circuits on the IBM's Athens topology, which is a linear connectivity, and map 9- and 10-qubit circuits on a 2D lattice of physical qubits. As discussed in Section 3.3.2, we compare industrial compilers such as IBM Qiskit [48] and $t|ket\rangle$ compilers [160]. Since the $t|ket\rangle$ compiler is shown to generate shorter circuits for several quantum applications [47], we use the circuits optimized by $t|ket\rangle$ compiler as our baseline. The circuits optimized using our Artisan with 3-qubit blocks and 4-qubit blocks are denoted as Artisan-3 and Artisan-4, respectively. Figure 3.6 shows the total number of CNOTs in the circuits optimized by the baseline and Artisan. Our Artisan optimization reduces the total number of CNOTs. For MUL5 and HLF5, Artisan-3 does not reduce the CNOT count because there is only a small number of CNOTs in a block. Artisan-4 can reduce the CNOT counts across all benchmarks. The average CNOT reduction of Artisan-4 is 29.9%. In general, Artisan-4 can achieve more CNOT reduction when compared with Artisan-3 because a large block will have more CNOTs in a block, and this is easier for the synthesis to find a circuit using less CNOTs than the original circuit block. Figure 3.7 shows the average CNOT count in a block. If a circuit is partitioned into 4-qubit blocks, the average CNOT count per block is higher when compared with 3-qubit blocks. As a result, Artisan-4 can achieve higher CNOT reduction. However, the time-to-solution of Artisan-4 is much longer than Artisan-3 (Figure 3.8).

3.5.2 Impact of Synthesis Distance

As discussed in Section 3.3.4, since there is a synthesis distance due to numerical approximation, the final state is not exactly the same as the ideal state of the original circuit. In order to analyze the impact of synthesis distance, we use ideal simulation to obtain the final states of the original circuit and synthesized circuit, and calculate the state infidelity [136]. Figure 3.9 shows the state infidelity. Since the unitary matrix distance of 4-qubit circuit synthesis is fundamentally larger than 3-qubit circuit synthesis, Artisan-4 has slightly larger

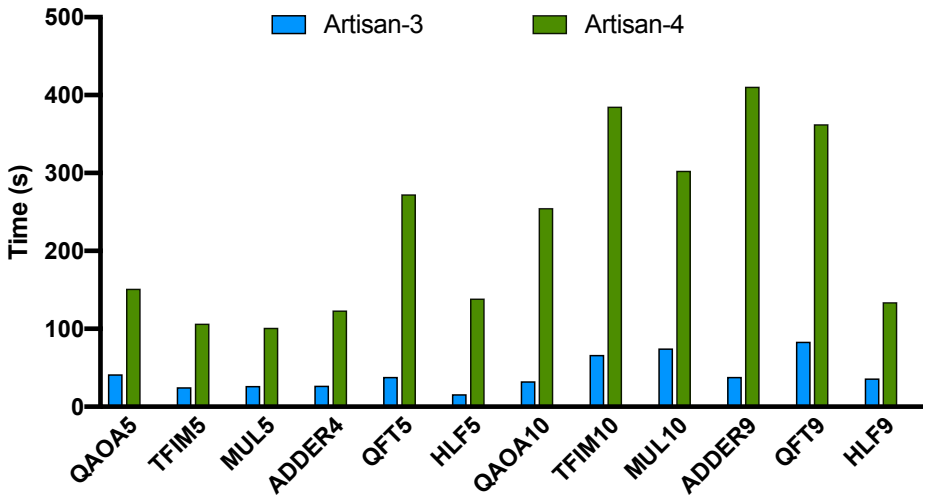


Figure 3.8: Compilation time.

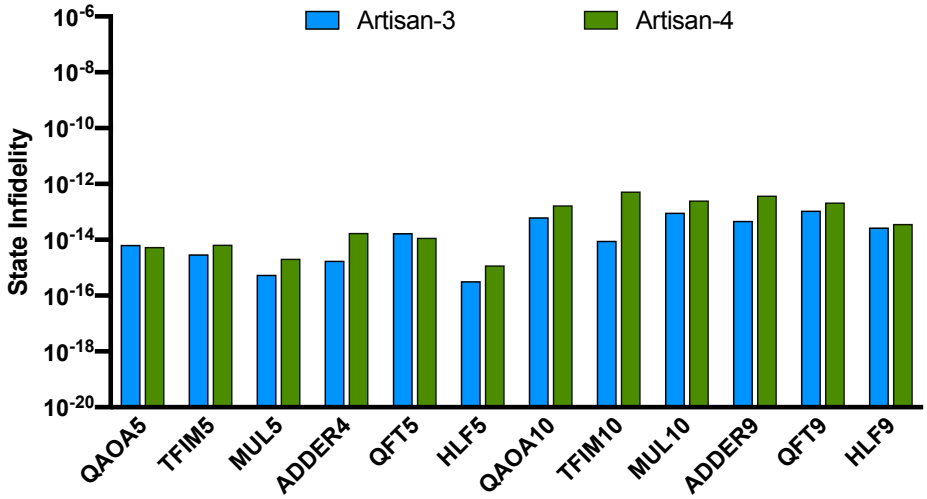


Figure 3.9: State infidelity due to the synthesis distance. Compared with the gate error on NISQ devices, the state infidelity due to synthesis distance is negligible.

state infidelity when compared with Artisan-3. The state infidelity of each optimized circuit is less than 10^{-12}. This is insignificant when compared with the gate error in the NISQ era. The current gate infidelity on real devices is ranged from 10^{-1} to 10^{-6}. Thus, the impact of synthesis distance is negligible.

3.5.3 Running on Real Hardware

To measure how much improvement our Artisan technique can achieve on the current available NISQ machines, we perform our experiments on quantum devices. Since our medium-size and large-size circuits are too deep for the current quantum devices to generate meaningful results, we choose small-size circuits for this evaluation. We run the small-size benchmarks on IBM's Athens device [92]. We use total variation distance, d_{TV}, to compare measurement samples of the real device with those of the ideal simulation. Total variation distance is commonly used as a metric for QC experiments [10, 120, 8, 60]. Lower d_{TV} means the samples of the real device are closer to the ideal distribution. Figure 3.10 shows the results on IBM's Athens, a 5-qubit device; each data point is obtained from 8192 shots of the circuit execution. We observe that Artisan achieves lower total variation distance for all benchmarks compared with the baseline optimization, and the distance results are correlated to the number of CNOT reduction. Even though the synthesis distance with 4-qubit blocks is larger than with 3-qubit blocks, Artisan-4 can achieve lower d_{TV} due to more CNOT reduction. Figure 3.11 shows the correlation between CNOT reduction rate and d_{TV} improvement (Δd_{TV}) compared with the baseline. The correlation coefficient is 78%. The results show that our CNOT reductions are indicative of fidelity savings. The results suggest that our Artisan optimization technique is important in the NISQ era, and is applicable to the current quantum devices.

3.5.4 Running on Noise Simulation

To understand the performance of Artisan optimization for medium circuits, we run the experiments on IBM Qiskit Aer noise simulation [4] because the medium circuits are too deep to get meaningful results on the current available real devices. To validate the trends shown by our simulations, we run small circuits on both real devices and simulations to see the correlation between them. Figure 3.12 shows the results from noise simulation. Compared with the results on IBM's Athens (Figure 3.10), the d_{TV} improvements are correlated in both

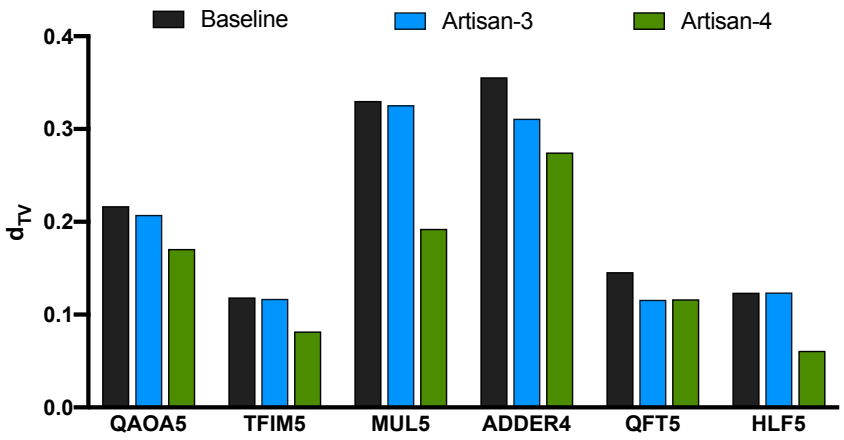


Figure 3.10: Running optimized small circuits on IBM's Athens, a 5-qubit device. (Lower d_{TV} is better.)

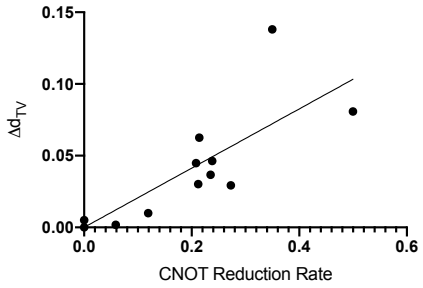


Figure 3.11: Correlation between CNOT reduction rate and d_{TV} improvement measured on IBM's Athens. Each data point is a benchmark from the small set. The correlation coefficient is 78%. The results show that our CNOT reductions are indicative of fidelity savings.

experiments. There is a 98% correlation between our real-device results and our simulation results on small circuits (Figure 3.13). For medium circuits on simulation, we show the correlation between CNOT reduction rate and d_{TV} improvement compared with the baseline in Figure 3.14. The correlation coefficient is 76%.

We perform sensitivity analysis by running noise simulations for medium circuits with different gate errors. Figure 3.15 shows the noise simulation results. We simulate depolarize_noise for two-qubit gate noises with the gate error probability from 0.1% to 2.5%. We can observe that Artisan can reduce more d_{TV} when the gate error is large.

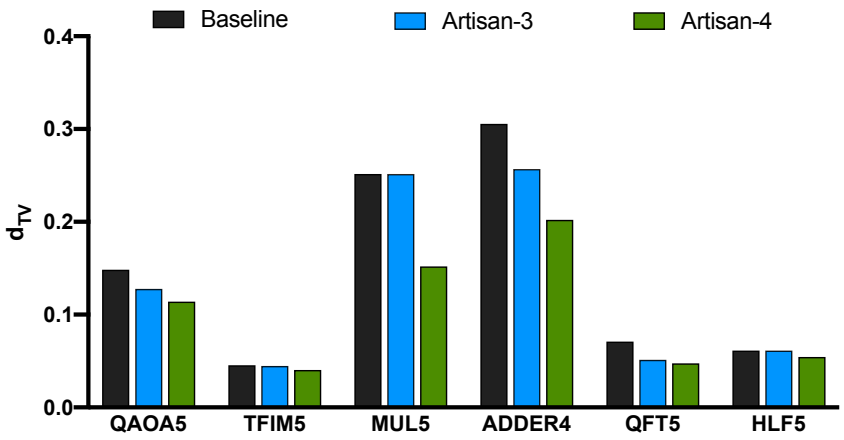


Figure 3.12: Running optimized small circuits on Qiskit noise simulation. (Lower d_{TV} is better.)

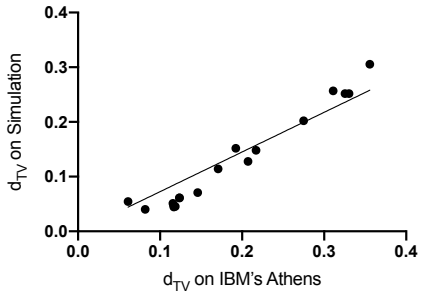


Figure 3.13: Correlation between d_{TV} results on the real device and noise simulation. Each data point is a benchmark from the small set. The correlation coefficient is 98%.

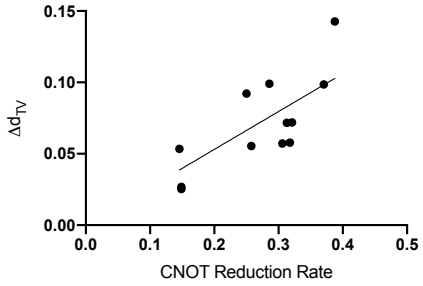
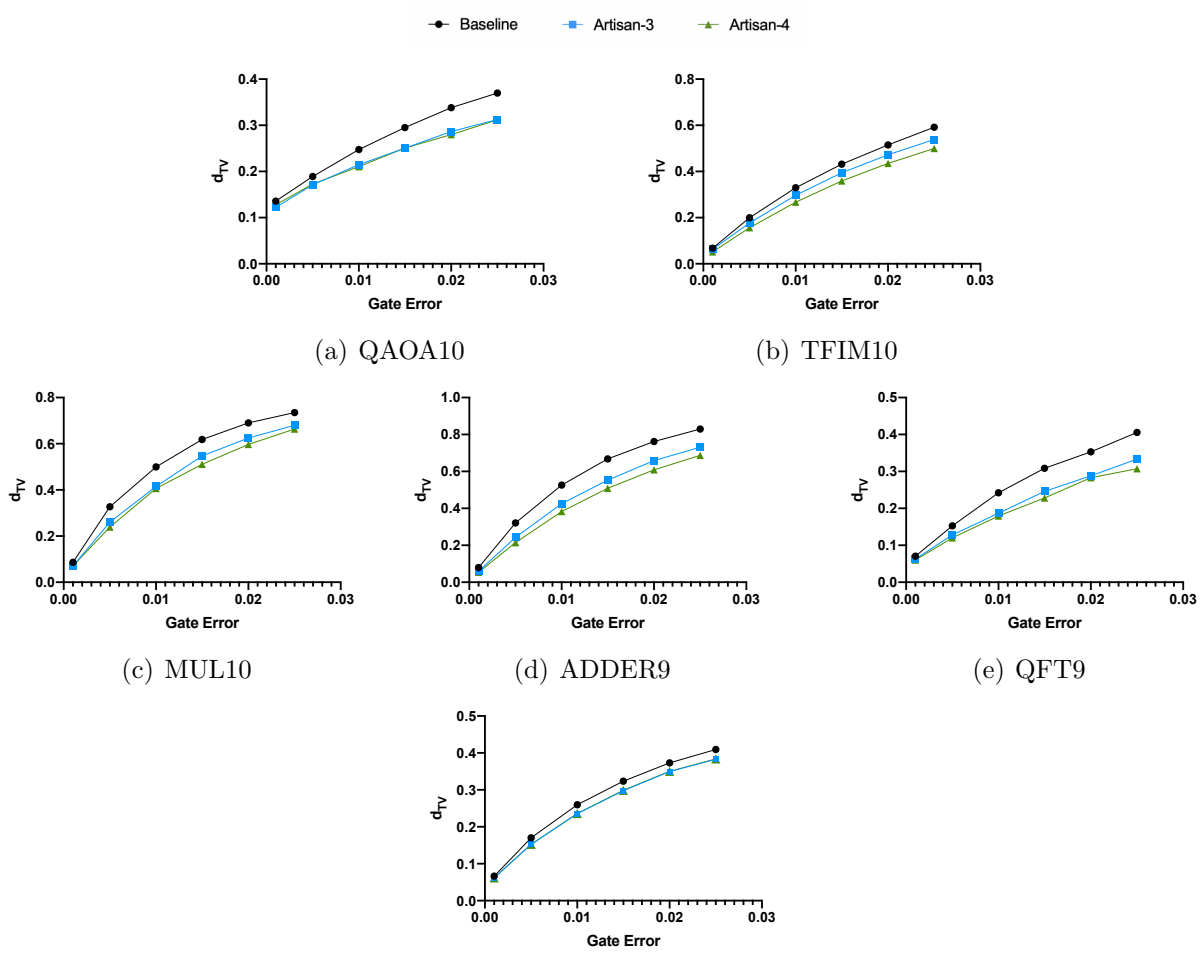


Figure 3.14: Correlation between CNOT reduction rate and d_{TV} improvement measured on noise simulation. Each data point is a benchmark from the medium set. The correlation coefficient is 76%.

3.5.5 Scalability

To demonstrate the scalability of our Artisan technique, we optimize the large-scale (60+ qubits) benchmarks using Artisan. We map the large-scale circuits on a 2D (8×8) lattice



(a) QAOA10 (b) TFIM10

(c) MUL10 (d) ADDER9 (e) QFT9

(f) HLF9

Figure 3.15: Realistic noise simulation results of d_{TV} under different levels of gate error. (Lower d_{TV} is better.)

of physical qubits. Figure 3.16 shows the total number of CNOTs optimized by the baseline compiler and Artisan, and Figure 3.18 shows the CNOT reduction rate. With large-size circuits, Artisan-3 and Artisan-4 achieves 22.8% and 28.9% CNOT reduction on average. Artisan-4 performs better than Artisan-3 in terms of CNOT reduction, but it takes longer time to complete the optimization. Figure 3.17 shows the compilation time of large-scale circuit optimization. Artisan-3 can complete the optimization within a few minutes, and Artisan-4 can finish the optimization process within a few hours.

Since the scale is too large to perform noise simulation, we use an analytical model to estimate the success rate of each circuit. The success rates in our evaluation are computed

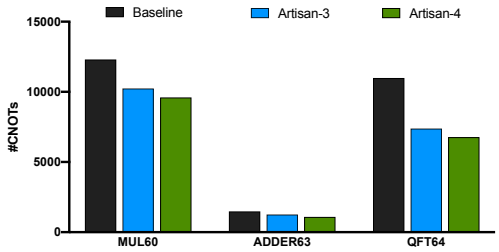


Figure 3.16: The number of CNOTs in the optimized large-scale circuits.

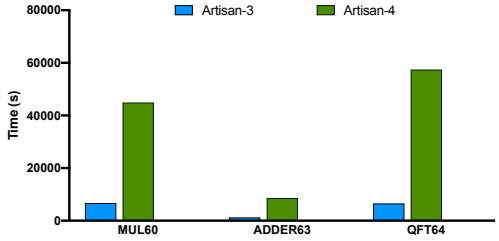


Figure 3.17: Compilation time of large-scale circuit optimization.

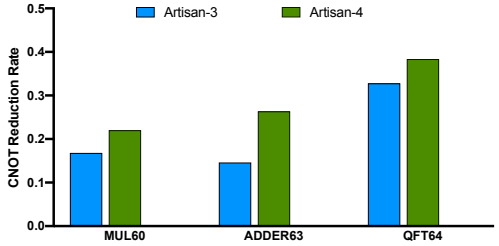
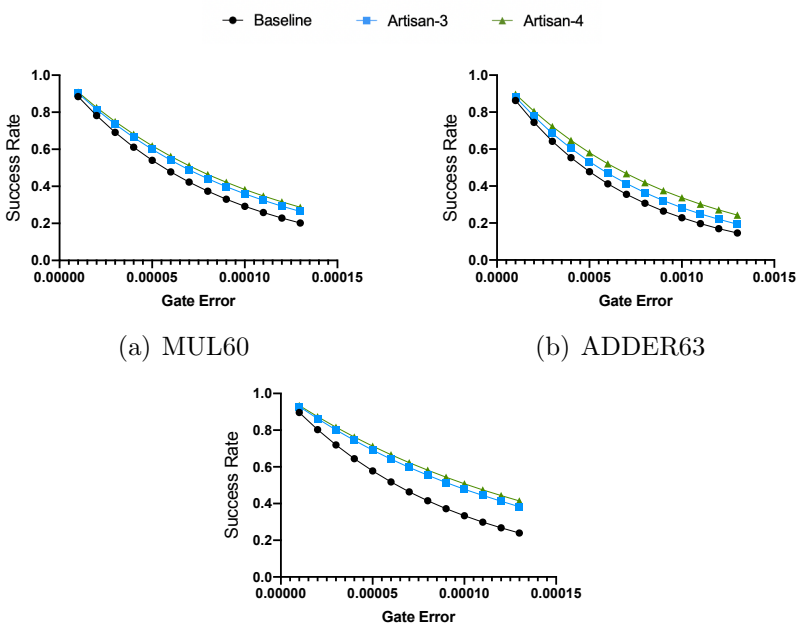


Figure 3.18: CNOT reduction rate of large-scale circuits.

by a worst-case analysis using gate success rates. Multiplying the gate success rates, we can obtain the estimated success rate for the whole circuit. Figure 3.19 shows the results under different gate error models. Since our Artisan-4 has the lowest CNOT count, it is projected to achieve the highest success rates for all benchmarks. The success rate improvement is greater when there is a larger gate error.

3.5.6 Discussion

The results show the general applicability of our approach. Having access to 3-qubit block synthesis already enables good optimization results on large circuits. In general, larger block size can achieve more CNOT reduction. Running synthesis with 5-qubit blocks is possible,



(a) MUL60

(b) ADDER63

(c) QFT64

Figure 3.19: Circuit success rate under different levels of gate errors. (Higher circuit success rate is better.)

but the solving time is much longer.

Artisan allows composability with any mapper available for a given platform and we have experimented with $t|ket\rangle$ compiler. Our preliminary sensitivity analysis of circuit quality to mapping quality indicates there is a direct correlation between Artisan efficacy and mapping quality. The best quality mappings have highly interconnected components. In our conjecture, the better mapper provides the higher opportunity of forming large blocks, thus motivating improvements in both mapping and synthesis. Our study offers insights for future compiler design.

3.6 Related Work

Several studies of circuit optimization have been carried out. Most of the existing techniques focus on optimizing the qubit mapping and swap insertion to reduce circuit depth. One common approach is to describe the problem in a mathematical form, such as integer linear

programming, and then find the optimal solutions by using solvers [11, 175, 121, 126, 78]. This approach only works for small circuits since the time scaling is exponential. Another approach is to find the optimal solutions by using dynamic programming [158, 94]. However, since the solving time grows exponentially, this method only works for a handful of qubits. Recent studies propose using heuristic search algorithms to find good solutions to avoid long execution times [94, 110, 187, 5, 149, 174, 105, 21]. However, these approaches keep the original CNOT count and only reduce the additional swap count. Our synthesis approach can reduce both swap count and CNOT count used in the circuits.

Previous studies have applied synthesis technique to optimize some specific circuits such as classical reversible circuits [14, 78, 173, 6] or Clifford+T circuits [138, 137]. In this work, our approach is designed for general circuits. However, since we can easily change the core synthesis tool, these synthesis approaches can be integrated in our compiler framework to improve the optimization for these specific circuits.

Our approach relies on synthesis techniques that are able to produce extremely short circuits. Otherwise, it is unlikely that we will be able to see an improvement when resynthesizing sub-circuits. The KAK decomposition could be used for resynthesizing 2-qubit blocks [168]. We have seen improvement when using larger block sizes, so we use the search-based technique found in [51], which can handle circuits as large as 4 qubits. For scaling further, we are considering the approach found in [182], which produces slightly longer circuits, but offers a better scaling runtime when compared to the search-based approach.

Recent studies such as [27] propose the removal of the constraint of unitary operations by adding ancilla qubits. Additionally, [39] uses ancillas and an approximation technique to produce very short circuits. To achieve greater CNOT reduction, integrating ancillas and approximate synthesis into our Artisan is a promising research direction.

3.7 Conclusion

In the NISQ era, since two-qubit gates are much noisier than single-qubit gates, it is essential to minimize their count. Synthesis is a powerful tool for circuit optimization to produce shorter circuits to improve the overall circuit fidelity. However, synthesis is only applicable for small circuits. In this work, we present an automated compilation framework, Artisan. It partitions the circuit into blocks, and re-generates each optimized block by using synthesis, and re-composes the circuit by stitching all the blocks together. Our approach to circuit optimization offers a role for quantum synthesis algorithms in large-scale quantum computing scenarios. We evaluate fidelity improvements using 3 metrics for 3 scaling regimes: small-size circuits using fidelity on real devices, medium-size circuits using fidelity using simulations with noise, and large-size circuits using CNOT reduction measured statically by our compiler. The results show that our technique has practical value on current devices and is reliable in the NISQ era. We also discuss using approximate synthesis to further trade for circuit depth and pulse-level optimization. Our study of circuit optimization using synthesis offers insights for future compiler design.

CHAPTER 4

TILT: ACHIEVING HIGHER FIDELITY ON A TRAPPED-ION LINEAR-TAPE QUANTUM COMPUTING ARCHITECTURE

Quantum computing (QC) aims to solve certain computational problems beyond the capabilities of even the largest classical high-performance computers. By leveraging the quantum mechanical principles of superposition and entanglement, QC algorithms have the potential to revolutionize areas such as machine learning [23], quantum chemistry [143, 98], and cryptography [156]. Recently, IBM, Rigetti, and Google demonstrated their QC devices up to 72 superconducting transmon qubits [91, 73, 147], while IonQ and Honeywell have recently made significant steps with trapped-ion QC devices [144, 93]. Current QC machines in the Noisy Intermediate-Scale Quantum (NISQ) era [145] are too small for large benchmarks and unable to support quantum error correction (QEC) [18, 76]. We can, however, run on the order of hundreds of quantum operations using on the order of hundreds of quantum bits (qubits).

Trapped-ion technologies are among the most promising systems for scalable QC for many reasons. While most current technologies have limited hardware qubit connectivity, two-qubit gates on trapped-ion computers can be executed on arbitrary pairs of qubits [80]. In a trapped-ion quantum computer, the internal states of the ions form the qubit states and quantum gates are implemented through laser-based operations [80, 178, 37, 114], where acousto-optic modulators (AOMs) apply lasers with carefully modulated amplitudes and frequencies in order to generate different quantum gates. In many trapped-ion technologies, the ions themselves must be physically moved, for example, shuttled across the linear trap storing all ions. Previous studies have shown that this transportation in a linear array can be done with minimal energy gain and without the loss of qubit coherence [148, 88, 17, 84, 163, 100, 24].

To scale the QC device, we can add ions to the chain, without fear of frequency collision

since all ions are identical. While adding ions is relatively simple, controlling all the ions simultaneously is challenging. Recently, a linear-tape architecture was suggested by Monroe [131], where the device had a fixed set of lasers (often much smaller than the total number of ions in the trap) used to operate all qubits. While the lasers are fixed in place, the ions can be shuttled across the tape to allow them to be operated on. Figure 4.1 illustrates this concept, where the *tape* is a string of ions (each representing a qubit) arranged in a single linear trap and the *tape head* is the set of control lasers used to manipulate the qubits. The location of the tape head designates the *execution zone*, or the set of locations where qubits must be located in order for gates to be executed. If a qubit is not located in the execution zone, before applying a gate we must physically move the ions until they are aligned with the tape head. Since the number of ions in the execution zone is limited at one time, the entire ion chain will be moved back and forth during the execution of quantum circuits, similar to a Turing machine [83]. While the linear-tape model is inefficient in classical computing, our evaluation shows that a quantum version, with modest parallelism in the tape head, is competitive with contemporary 2D quantum computing designs. The key is that the qubits under the tape head form a completely connected graph and that this powerful communication structure can slide across the entire tape. Shuttling a small chain of ions has been demonstrated in [64], and larger tape-like demonstrations are planned [34].

By sharing laser controls, a trapped-ion linear-tape (TILT) architecture has easier calibration and simplified optics, as well as reduced hardware costs. In a TILT system, gates are implemented only on the qubits within the execution zone, and the entire chain shifts back and forth to operate all qubits. Such a machine does not require the difficult shuttling primitives of a quantum charge-coupled device (QCCD) architecture [102], since full chain shuttling is not nearly as difficult as split/merge operations or shuttling over junctions. Moreover, since the ions in the center of a trap are more evenly spaced (which can be accentuated by trap design), such an architecture has fewer issues with individual addressing and laser pointing errors. Consequently, gate fidelities would improve in a TILT architecture,

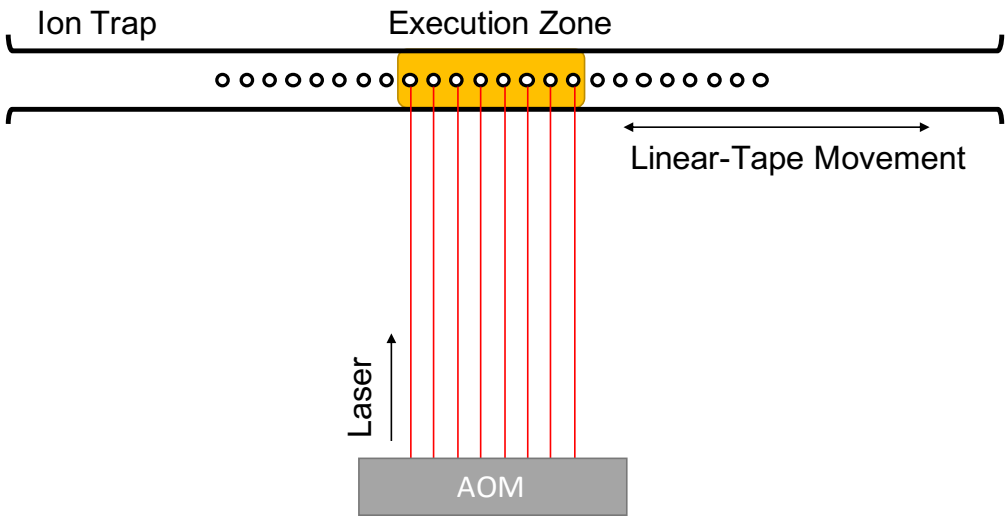


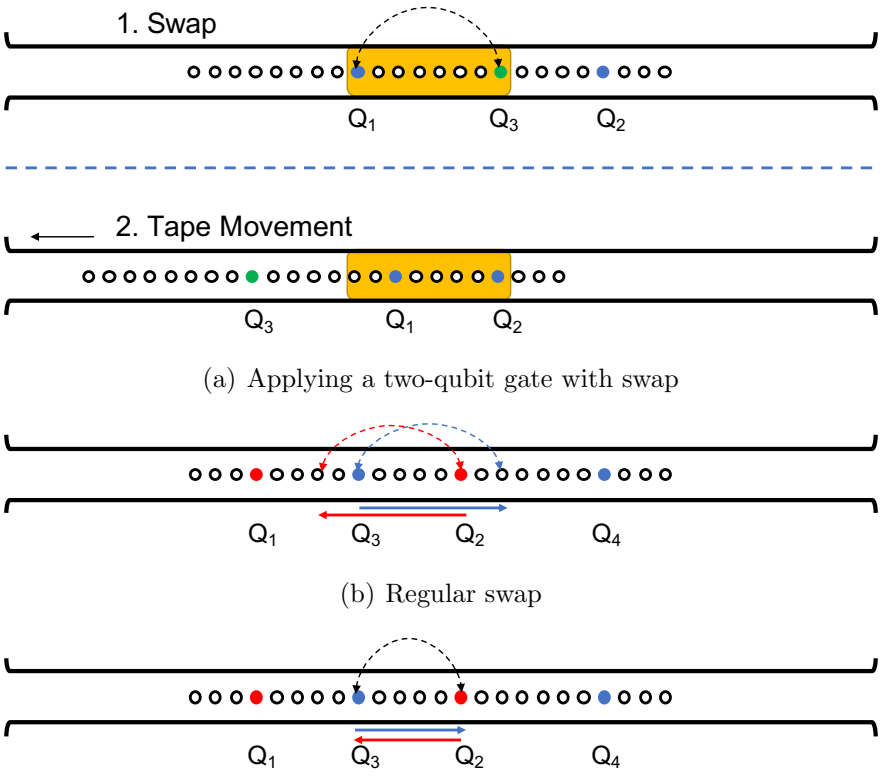
Figure 4.1: A trapped-ion linear-tape quantum computing architecture. Acousto-optic modulators (AOMs) target laser beams for quantum operations, which can be applied to ions in the execution zone. In order to perform gate operations on the other qubits, the entire ion chain is translated until the target qubit is moved into the execution zone.

and control optics would be required only for a much smaller region of the chain, resulting in reduced control complexity and cost.

The construction of trapped-ion quantum computers is heavily influenced by commodity technology, just as classical computing architectures have been. In particular, trapped-ion systems exploit lasers and AOMs used for writing chip masks for commodity silicon chips. The frequency of these commodity lasers determines the ions used (Ytterbium); but more significantly, the size of the AOMs limits the size of the control head to 32 lasers.

In a TILT system, since the tape head is not covering the entire ion chain, full connectivity is no longer supported among all qubits. For a two-qubit gate, if the distance between the two involved qubits is less than or equal to the tape head size, this two-qubit gate is executable. It may require the whole chain to be moved under the head, but no other gates are required. However, if the distance between the pair of qubits is more than the tape head size, both swap gates and tape movements are necessary, as in Figure 4.2(a).

A linear architecture makes communication via qubit swaps more efficient by channeling movement in one path, forcing data to pass in opposite directions; and requiring fewer swap



(a) Applying a two-qubit gate with swap

(b) Regular swap

(c) Opposing swap

Figure 4.2: (a) A two-qubit gate is applied to Q_1 and Q_2 (marked by blue). Since the distance between Q_1 and Q_2 is longer than the execution zone, a swap between Q_1 and Q_3 (marked by green) is needed. After the swap, a tape movement brings Q_1 and Q_2 in the execution zone for the two-qubit gate execution. (b) Suppose a two-qubit gate is applied to $\{Q_1, Q_2\}$ (marked by red) and another two-qubit gate is applied to $\{Q_3, Q_4\}$ (marked by blue). Two regular swaps are needed to make the two-qubit gates executable. (c) An opposing swap combines two regular swaps in opposite directions, and this swap makes both two-qubit gates executable. Thus, creating opposing swaps can reduce the total number of swaps.

gates involved in the program execution, achieving higher circuit fidelity. Utilizing this architectural feature, we can frequently pair up regular swaps to create *opposing swaps*. As Figure 4.2(c) shows, an opposing swap combines two swaps in opposite directions, and hence each opposing swap is equivalent to two regular swaps. This effect, combined with modest tape head sizes, can substantially reduce swaps as compared with 2D architectures.

Regarding the ion chain movement, our scheduling objective is to maximize the overall quantum program success rate. Previous noise analysis studies showed that motional excita-

tions during shuttling are strongest when the acceleration of an ion is the highest [181, 170]. As a result, the distance traveled, which is generally traversed at a nearly constant velocity, is not particularly relevant to the heating. Instead, there is a chance of heating when the ion begins or ends its motion. When the ion chain starts and stops moving, the movement will trigger the possibility of an increase in thermal motion. This thermal motion can introduce errors during the application of multiqubit gates, as discussed in Section 4.3.5. Thus, minimizing the number of movements can improve the overall circuit success rate.

To support our analysis on TILT, we have developed *LinQ*, a toolflow to compile from high-level quantum programs to a TILT architecture that minimizes the number of swaps and tape moves and hence improves the expected success rate. We implement a simulator for TILT to evaluate application metrics by using a suite of NISQ applications. We demonstrate that the TILT architecture outperforms QCCD architectures in terms of program success rate in a range of NISQ benchmarks.

The main contributions of this work are summarized as follows.

- To evaluate the feasibility of TILT systems, we present, for the first time, a comprehensive design and evaluation of the TILT architecture targeting systems with 64 qubits.

- To evaluate the performance of TILT architectures, we develop our toolflow, LinQ, that provides an optimizing compiler and simulator. We develop swap insertion and shuttling strategies to improve the overall quantum program success rates.

- To precisely understand the impact of thermal heating, we simulate using a noise model extracted from realistic experimental studies to estimate the application reliability.

- Our simulations show that TILT can achieve higher program success rates on a range of NISQ benchmarks compared to QCCD systems (up to 4.35x and 1.95x on average). Our results suggest that TILT provides a viable path toward scalable quantum computers.

4.1 Background

In this section, we give a brief overview of quantum computation. We then present the relevant background on trapped-ion systems.

4.1.1 Principles of Quantum Computation

Qubit. A qubit is a two-level quantum system usually defined by two computational orthonormal basis states $|0\rangle$ and $|1\rangle$. A quantum state can be expressed by any linear combination of $|0\rangle$ and $|1\rangle$: $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, where α and β are complex amplitudes satisfying $|\alpha|^2 + |\beta|^2 = 1$.

Quantum Gates. Quantum gates are unitary operations applied on qubits to map from one quantum state to another. Gates are applied to one or multiple qubits simultaneously. Arbitrary single-qubit gates and two-qubit controlled gates are known to be universal [61]. A quantum circuit (application) consists of a sequence of quantum gates to evolve the quantum state toward the solution state.

4.1.2 Trapped-Ion QC Systems

In a trapped-ion quantum computer, information is stored in the internal states of atomic ions which are trapped in an oscillatory radio-frequency magnetic field. This field constrains the positions of the ions, and almost all current proposals have the ions arranged in a linear crystal, commonly referred to as the *ion chain*. The internal states of the ions can be manipulated by shining lasers on individual ions in order to implement single-qubit gates, and the motion of the chain is used as a bus in order to mediate entangling operations between multiple qubits. The qubit can be defined by either the hyperfine states or Zeeman states of the ion, each of which has benefits beyond the scope of this study [36].

Ion Chain and Qubit Characteristics. The ions in most modern ion traps are spaced approximately 5 microns apart. We do not consider T_1 or T_2 times in this study, but for

a qubit built off of the hyperfine states of Yb_+^{171} ions, the former is on the order of $10^{11}s$ and the latter is generally on the order of seconds, but measurements of up to ten minutes have been achieved [171]. The state of the qubits in this chain can be measured by shining a laser on the beam, which causes ions in the $|1\rangle$ state to fluoresce. The ion crystal can then be imaged, with ions in the $|1\rangle$ state appearing as bright spots, and ions in the $|0\rangle$ state appearing as dark gaps [140].

Laser-Based Operation. In a trapped-ion quantum computer, gates are applied via laser pulses aimed at the target ions. Different operations can be implemented by varying the intensity, frequency, phase, or length of these pulses. For example, consider a single qubit gate. A laser is aimed at the ion which causes its state to slowly rotate, in the Bloch sphere, about the desired Pauli axis. The rotation speed can be increased by increasing the intensity of the beam. By carefully monitoring the intensity of the laser and tracking the timing, the desired rotation can be implemented. Two qubit gates are more complex; for trapped-ion quantum computers, the canonical two-qubit gate is the Mølmer-Sørensen gate, which implements an $XX(\theta) = \exp(i\frac{\theta}{2}\mathbf{XX})$ operation by using lasers individually addressed on the desired pair of qubits, along with a global beam that hits all qubits in the chain. In order to create long-range two-qubit entanglement, this gate entangles each qubit with the motional state, applies state-dependent forces, and then unentangles the qubits from the motion. If done perfectly, this process leads to ions whose internal states are entangled without any residual entanglement between the internal states and the motion. In modern experiments, raw single-qubit gates have error rates around 10^{-3}, although by using composite pulses these rates can be improved significantly [35, 176]. Error rates on two-qubit gates often depend on the length of the chain; but in small experiments, numbers as low as 10^{-3} have been reported, although most experiments are still on the order of 10^{-2} [109, 13, 178].

Tape Shuttling. In the proposed TILT architecture, the physical ion chain takes the place of the linear tape. By modulating the DC voltages of the electrodes in the trap, the ion chain can be moved along the axis of the trap. As explained above, lasers are used to

process the information in a trapped-ion quantum computer. The idea behind an TILT architecture is that operations are applied only to ions near the middle of the trap, in the *execution zone*, and the chain is shifted back and forth in order to make long-range interactions occur. This strategy is counter to another architectural proposal, the quantum charge-coupled device [102]. A QCCD architecture comprises many smaller trapping zones within a single chip, and ions are shuttled around individually. Recently, Honeywell built the first QCCD system with four qubits [144]. While such an architecture allows for more parallelization and flexibility, it has high costs in terms of shuttling complexity since it requires expensive split/merge and junction crossing maneuvers, which lead to more thermal energy entering the system.

Thermal energy is stored in the motion of the chain, and it is also a significant factor in TILT architectures. The Mølmer-Sørensen gate mentioned in the preceding section is designed to be insensitive to the motional energy of the chain; however, this is true only for a perfectly applied gate. In reality, there is a small contribution to the infidelity of the gate, which is caused by residual entanglement between the internal state of the ions and the motional state, which was used to mediate the interaction. This error is due to the imperfect closure of a loop in phase space (the space of the positions and momenta of the particle). The error caused by this imperfect closure scales as $\exp(2n + 1)$, where n is the number of motional quanta in the chain. As a result, as a chain heats up, it becomes more sensitive to imperfections in the application of the gate.

4.2 TILT Architecture

TILT architectures have a linear chain of ions trapped within the oscillating potential which shuttles as one large block. We call this linear chain a *tape*. Most shuttling studies focus on split and merge operations or shuttling single qubits through junctions. A primary benefit of the linear tape architecture is that these difficult maneuvers are unnecessary.

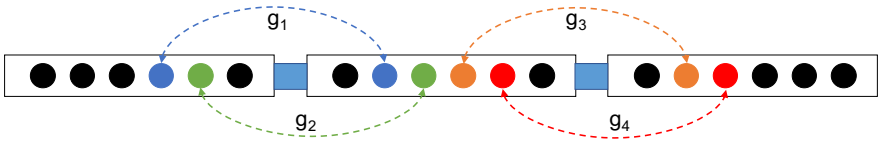
4.2.1 Feasibility

Prior works [181, 170] show that shuttling distance of ions is irrelevant to the heating since ions are shuttled at a nearly constant velocity. However, the start and stop of the ion shuttling will add thermal heating to the ion chain. Based on previous noise analysis studies [181], we derive a heating model in which every time the chain is shuttled its motional energy increases by some fixed amount k. This quantity is affected by how free the chain is to oscillate in its center of mass mode after shuttling. As a result, we scale this value by $k \sim \sqrt{n}$ when moving to a larger chain, where n is the number of ions of the chain. Whenever a two-qubit gate is enacted, its fidelity is dependent on the total motional energy of the chain. High motional energy leads to larger gate errors, so shuttles have a negative impact on the overall success rate of the future gates. As a result, compiling techniques minimizing the number of tape moves are favored.

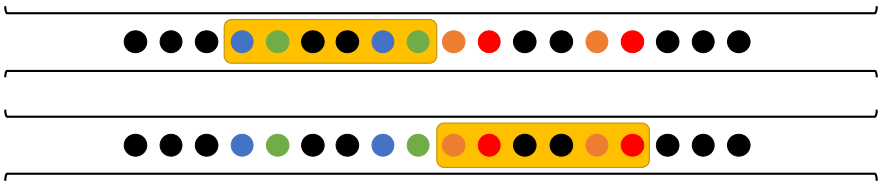
All components of this architecture have been developed to some extent, however ion trap quantum computing experiments have yet to be attempted at the point where our techniques are necessary. Chains as large as 79 ions have been demonstrated by ionQ [93], however they didn't individually address all ions. Using a TILT architecture would make this addressability concern much simpler. Shuttling techniques like the ones needed for our work have already been demonstrated for small segments of crystals, and would be scaled up to larger chains without a increase in complexity [144, 170, 81]. The benefit of the TILT architecture is that it does not require any pieces that have not been demonstrated already at the level of complexity necessary for the architecture, as opposed to QCCD architectures which will need more complex junction shuttles and trap designs as they scale up from their small demonstrations to larger quantum devices.

4.2.2 Gate Selection

There are multiple gate implementations for trapped-ion systems, including Frequency Modulated (FM), Phase Modulated (PM), and Amplitude Modulated (AM) gates [133, 178,



(a) Qubits communication in QCCD.



(b) Qubits communication in TILT.

Figure 4.3: Compare TILT with QCCD. g1, g2, g3, and g4 are two-qubit gates on four pairs of qubits (marked by blue, green, orange, and red). (a) For QCCD, when a qubit communicates with another qubit in the different trap, QCCD requires the following operations: i) Swap the qubit to the end of the current trap. ii) Split from the current chain. iii) Shuttle to the target trap. iv) Merge to the target chain. v) Interact with the target qubit. To execute the gates g1, g2, g3, and g4, QCCD needs to repeat the above operations four times. (b) For TILT, performing a tape movement would allow the operations of the desired two-qubit gates.

26, 12]. The gate time for FM gates is proportional to the total number of ions in a chain [109, 108]. However, the operation time of PM and AM gates is only proportional to the distance between two involved ions [181, 167, 128]. Since TILT has a long chain with a smaller region for operations, using FM gates would not take advantage of our systems structure, leading to extra gate error. Therefore, PM or AM gates are proper gate implementations for TILT. In our study, we consider amplitude modulated (AM) gates for our two-qubit gate implementation.

4.2.3 Compared with QCCD

Another proposed model to scale trapped-ion QC systems is the quantum charge coupled device (QCCD), which requires complex junction shuttles and trap designs as they scale up [102]. Additionally, QCCD requires ion chain split/merge operations during the process of a computation [133] for communication with qubits in other traps.

Figure 4.3 shows the operations required for a short-distance (within the laser head size)

communication pattern. QCCD requires a series of operations to perform all two-qubit interactions. However, TILT only needs a single tape movement to achieve the operations. As a result, TILT is expected to have higher success rates. Furthermore, if there are multiple short-distance communications, a QCCD system would require multiple split/merge/shuttle operations, but it is possible for TILT to achieve these operations with only one tape movement through a well-designed tape move scheduling. In this case, TILT will outperform QCCD architectures. For long-distance communication, QCCD requires almost the same operations for short-distance communication if there is a shuttling pathway, but for TILT, the system might require multiple swaps and tape shuttles. Thus, QCCD can perform better than TILT in dealing with a long-distance communication.

In the current state of QC, most quantum applications heavily rely on short-distance communication patterns, such as the Variational Quantum Eigensolver (VQE) [98], Quantum Approximate Optimization Algorithm (QAOA) [65], Quantum Error Correction (QEC) like the surface code [66], Quantum Generative Adversarial Networks (QGAN) [50, 118, 159, 86, 183], and the Ising model solver [16]. Therefore, TILT is a promising design to achieve higher fidelity on these applications with a trapped-ion quantum computing architecture.

4.2.4 TILT Challenges

Under the device limitations of TILT, there are two major challenges to performing quantum circuits. Since the qubits are not fully connected, two-qubit gates on pairs of qubits whose distance is greater than the head size will need swap gates in order to make two qubits close enough to become executable. Consequently, we need qubit mapping and swap insertion techniques which minimize the number of swaps in order to achieve high quantum program success rates. Second, since only a part of the ion chain can be moved into the execution zone and since the tape movement will cause a heating error, minimizing the number of tape moves is essential to success.

4.3 LinQ for TILT Architecture

To evaluate the TILT architecture, we develop our toolflow, LinQ, including the compiler and simulator. LinQ allows us to compile a quantum program written in a high-level programming language to TILT-architecture-level instructions. LinQ takes the device specification as input, including the number of qubits and the tape head size, and returns a compiled circuit, optimizing for program success rate. Since the TILT machine is not yet realized, we use simulation to predict the performance of this novel architecture. We summarize the notations used in this paper in Table 4.1.

Table 4.1: Notations used in this paper.

Notation	Definition
n	The number of ions of a chain.
g	A two-qubit gate.
$g.q_1, g.q_2$	A pair of qubits that g is applied to.
d_g	The distance between $g.q_1$ and $g.q_2$
M	A mapping from logical to physical
M_{q_i,q_j}	A qubit mapping after swapping (q_i, q_j)
G	A set of two-qubit gates
$D(g, M_{q_i,q_j})$	d_g under the mapping M_{q_i,q_j}
Γ	Background heating rate of the trap
τ	Gate time
k	Amount of heating added during each shuttle
ϵ	Gate error due to residual entanglement

4.3.1 LinQ Overview

Figure 4.4 shows an overview of our toolflow. LinQ contains three core compiler modules: trapped-ion native gate decomposition, qubit mapping and swap insertion, and tape movement scheduling. Since our target device size for TILT is more than 60 qubits, using one procedure for optimizing qubit mapping, swap insertion, and tape movement scheduling will have a long compilation time. Therefore, we separate the optimization problem into two steps and address them independently. However, when we perform swap insertion, we should take tape movement scheduling into consideration, because the swap patterns will affect the

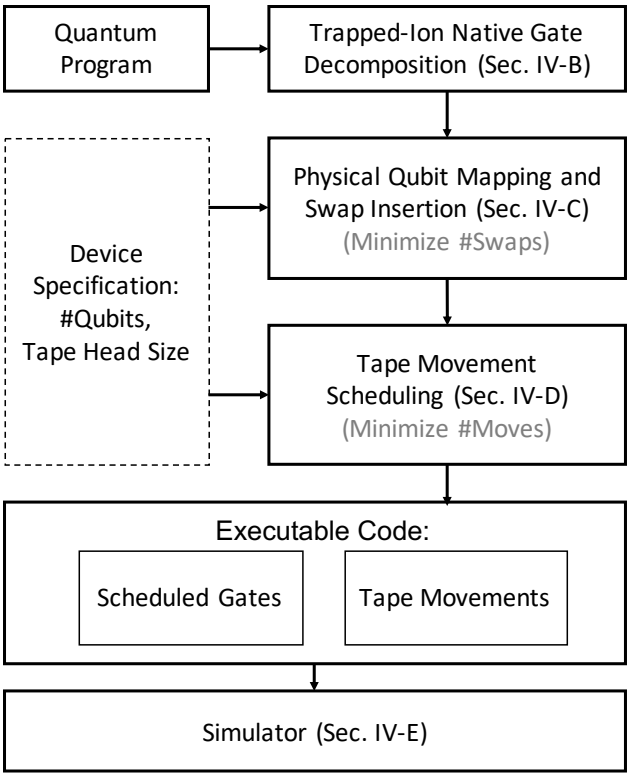


Figure 4.4: LinQ overview. Taking a quantum program and device specification as input, LinQ generates the executable codes, including a sequence of scheduled gates and tape movements, and then runs the simulation to output the success rate and run-time.

number of tape movements. In Section 4.5.2 we will show that this two-step optimization can generate success rates similar to those of ideal trapped-ion systems.

The inputs for LinQ consist of a high-level quantum program and the device specification, including the number of qubits in the tape and the tape head size. First, LinQ decomposes the quantum program to trapped-ion native gates. Next, LinQ maps the logical qubits to physical qubits and resolves all long-distance two-qubit gates by inserting swaps gates. Then, LinQ performs our tape movement scheduling algorithm. The output is the executable circuit consisting of a sequence of scheduled gates and tape movements.

To evaluate the performance, such as success rates and the execution time, we pass the sequence of scheduled gates and tape movements to our simulator, which uses noise models extracted from realistic experimental studies.

4.3.2 Native Gate Decomposition

In order to generate executable code, gates from high-level quantum programming languages must be decomposed into the device-dependent native gates. LinQ decomposes the universal gates, such as CNOTs, CZs, and other operations into the TILT native gate set [125]. For example, we decompose a CNOT q_1, q_2 gate into a sequence of rotations and XX operations: $R_y(\pi/2)\ q_1$; $XX(\pi/4)\ q_1, q_2$; $R_x(-\pi/2)\ q_1$; $R_x(-\pi/2)\ q_2$; $R_y(-\pi/2)\ q_1$.

4.3.3 Qubit Mapping and Swap Insertion

Unlike fully connected trapped-ion systems, TILTsystems cannot operate long-distance two-qubit gates directly if the distance between the pair of qubits is longer than the tape head size. These long-distance two-qubit gates become unexecutable gates and to resolve them, we must insert swap gates. For a two-qubit gate g applied on q_1 and q_2, we use d_g to denote the distance between q_1 and q_2. The swap insertion problem is to find a sequence of intermediate qubits, for example, $\{(q_1, q_i), (q_i, q_j), \ldots, (q_k, q_2)\}$, such that any pair (q_i, q_j) in the sequence has distance smaller than the tape head size.

Several approaches of swap insertion have been proposed. One common approach is to formulate the problems in a mathematical form, such as integer linear programming, and then utilize software solvers to find the optimal solutions [11, 175, 121, 126, 180]. This method is guaranteed to provide an optimal solution. However, since the time-to-solution grows exponentially with the number of qubits and the circuit depth (or the number gates), scaling this approach to large NISQ programs is infeasible. In our study, we focus on 60+ qubits applications. Hence, we need a practical solution for the swap insertion problem.

Baseline Approach. A straightforward idea is to allow swap with the tape head size distance to ensure a minimal number of swaps required for a two-qubit gate. We firstly establish our baseline implementation by using IBM Qiskit *StochasticSwap* [48], which is commonly used for swap insertion and it is also used in the highest optimization level of Qiskit circuit compilation pass.

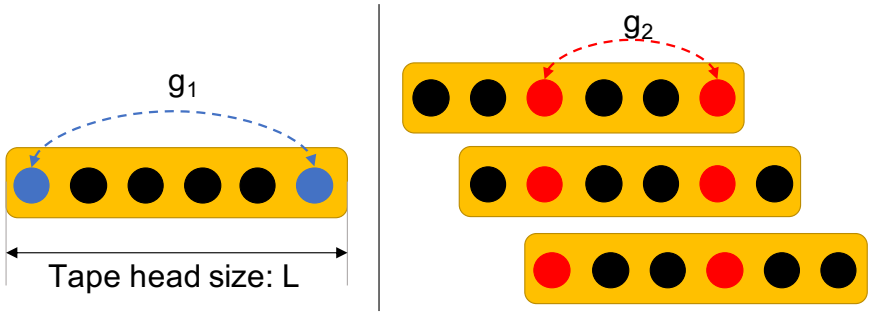


Figure 4.5: Considering two-qubit gates g_1 and g_2 on a system with the tape head size of L. Since d_{g_1} is $L-1$, only one tape head position is allowed for g_1 to be executed. As for g_2, there are three valid tape head positions because d_{g_2} is $L-3$.

However, two problems arise. First, using the tape head size as the swap distance will force the tape to move exactly once for every swap. If d_g is less than the tape head size, the tape movement scheduler will have multiple choices to schedule the tape position for gate g; If d_g is equal to the tape head size, the scheduler must move the tape to the exact position for the two-qubit gate, and hence this will increase the number of tape movements in many cases. Second, this swap insertion method does not consider opposing swaps. An opposing swap (Figure 4.2(c)) combines two regular swaps in opposite directions. As a result, creating opposing swaps frequently can reduce the total number of swaps. This reduced swap count results in greater circuit success rates.

A good qubit mapping algorithm can also reduce the number of swap gates. Previous studies have proposed two types of solutions for qubit mapping and swap insertion for 1D and 2D architectures. One is to formulate the problem as an equivalent mathematical problem and then use a solver to find the optimal mapping to the problem [126, 40]. This approach cannot scale to solve large problem for similar reasons to the swap insertion problem. The other solution is to use a heuristic search to obtain a mapping [139, 110, 94]. Since our target size is 64 qubits, we use a heuristic search to address this problem.

LinQ Approach. We adopt the existing heuristic algorithms for initial qubit mapping [110, 94]. After we have an initial qubit mapping M, we improve the swap insertion. For resolving an unexecutable two-qubit gate g on q_1 and q_2, we insert swaps by applying our

heuristic algorithm, shown in Algorithm 2. The idea is to create opposing swaps to reduce the overall swap count. For each two-qubit gate applied on q_1 and q_2, for all q_i between q_1 and q_2, if the distance between (q_i, q_1) or (q_i, q_2) is less than the maximal swap length ($MaxSwapLen$), then we put the swap candidate into a list S. Next, we compute the score for each swap candidate and select the swap with the minimal score to update the mapping and insert the swap to the circuit. If the tape head size is L, $MaxSwapLen$ can be $L-1$. For a swap gate g, if d_g is equal to $L-1$, the tape must be moved to the exact position for the gate execution, as shown in Figure 4.5. As a result, the circuit might require more tape movements. To mitigate this problem, we restrict $MaxSwapLen$ to a certain number less than $L-1$ to create the flexibility for our tape movement scheduling. As long as the swap gate count is not increased significantly, by shorting the $MaxSwapLen$, we can increase the available head positions to potentially operate more gates in one tape movement. The best choice of the $MaxSwapLen$ depends on the structure of the target application. An ideal $MaxSwapLen$ will limit the overall d_g but will not increase the total number of swap gates. We can repeatedly run the procedure with different parameters to select the best $MaxSwapLen$.

The heuristic score function sums d_g for each two-qubit gate g under a mapping M. If a swap pair (q_i, q_j) leads to the minimal sum of all d_g, this swap pair will be selected. The swap score of the qubit pair (q_i, q_j) is defined as follows:

$$Score(M_{q_i,q_j}) = \sum_{g \in G} D(g, M_{q_i,q_j}) \times \alpha^{\Delta(g)}, \tag{4.1}$$

where G is the set of remaining two-qubit gates, M_{q_i,q_j} is the qubit mapping after swapping q_i and q_j, $D(g, M_{q_i,q_j})$ denotes d_g under the mapping M_{q_i,q_j}, α is a parameter $(0 < \alpha < 1)$ used to prioritize two qubit gates, and $\Delta(g)$ is the circuit depth distance between the gate g and the current resolved gate. We use the sum of d_g to consider the impact for the future swaps. The optimal swap can impact the swaps in the future to reduce the overall qubit

Algorithm 2 Swap Insertion

$S \leftarrow \phi$
while g is unexecutable **do**
    $C \leftarrow \phi$
    for q_i from $g.q_1$ to $g.q_2$ **do**
        if $dist(q_i, g.q_1) < MaxSwapLen$ **then**
            $C \leftarrow C \cup \{(q_i, g.q_1)\}$
        end if
        if $dist(q_i, g.q_2) < MaxSwapLen$ **then**
            $C \leftarrow C \cup \{(q_i, g.q_2)\}$
        end if
    end for
    for (q_i, q_j) in C **do**
        Compute $Scores(M_{q_i,q_j})$
    end for
    Find the swap with minimal score
    $S.append(Swap(q_i, q_j))$
    $M \leftarrow M_{q_i,q_j}$
end while
Insert the swap gate sequence S to the circuit

distance between the pair of qubits. Thus, we choose the swap candidates have the minimal scores.

The computational complexity of our heuristic algorithm can be estimated by the worst case, where the two qubits are separated in two ends and the swap insertion is performed repeatedly until two qubits are close enough to make the two-qubit gate become executable. The complexity is $O(NG)$ for a two-qubit gate, where N is the total number of qubits in a tape and G is the number of remaining two-qubit gates.

4.3.4 Tape Movement Scheduling

In TILT architectures, since the tape head is shared among all qubits, the tape will be moved back and forth frequently to execute a quantum application. Such tape movement/shuttling can introduce noise into the systems. Aware of issues like this, an effective tape movement scheduler can increase circuit fidelity by mitigating shuttling noise. In this work, we model the fidelity of the operations in order to improve the success rate. Our noise model considers

Algorithm 3 Tape Movement Scheduling

$T \leftarrow \phi$
$E \leftarrow \phi$
$R \leftarrow G$
while R is not empty **do**
    for each tape head position p_i **do**
        $F_i \leftarrow \{ExecutableGates\}$
        Compute $Score(p_i)$
    end for
    Find the tape head position p with the maximal score;
    $T.append(p)$
    $E.append(F_p)$
    $R \leftarrow R - F_p$
end while
Output the tape movement scheduling T and gate execution sequence E

the impact of thermal heating due to shuttling [181]. Since every tape move introduces thermal heating to the TILT system, this introduced heating leads to a lower gate fidelity. Hence, The design of our scheduling algorithm is to perform as many operations as possible before a tape movement is performed. Algorithm 3 shows the pseudocode of our scheduling algorithm. We compute the scores for each tape head position and select the position with the maximal score to schedule; we then repeat the procedure until all gates are scheduled and executed.

The objective of the heuristic cost function is to indicate the number of executable gates for a tape head position. As a result, the number of executable gates is the score for each tape head position. The general form of our heuristic cost function for a tape head position p is shown as follows:

$$Score(p) = n_p, \tag{4.2}$$

where n_p is the number of executable gates at the position p.

The time complexity of our heuristic scheduling algorithm is $O((N-L)DL)$, where N is the number of qubits in a trap, L is the tape head size, and D is the circuit depth.

4.3.5 TILT Simulation

To understand the impact of shuttling noise on TILT architectures, we perform noisy circuit simulation using realistic noise models. In our simulations we consider amplitude modulated (AM) gates for our two-qubit gate implementation [167]. These gates have a gate time of

$$\tau(d) = 38 \times d + 10 \tag{4.3}$$

where d is the distance between the two ions, in units of ion spacings, and the time is given in microseconds. This same model was studied in [133].

In an ion-trap quantum computer, single-qubit gate fidelities are independent of the thermal energy in the trap. Additionally, a perfectly applied Mølmer-Sørensen gate is independent of the thermal energy of the chain. As the vibrations increase, however, the gate becomes more sensitive to any imperfections in the laser pulses, which we quantify through the following model of gate fidelities after m moves have occurred, inspired by the work done in [181]:

$$F_m = 1 - \Gamma\tau + (1 - (1 + \epsilon)^{2mk+1}) \tag{4.4}$$

where Γ is the background heating rate of the trap, τ is the gate time defined in microseconds as a function of number of intra-ion spacings between the involved qubits [133], k is the average amount of heating added to the chain during each shuttle, as discussed in Section 4.2; and ϵ is the error in each two-qubit gate due to residual entanglement with the motion. A gate can be thought of as a loop in phase space, where a perfect gate is a closed loop. An imperfect gate, due to a bad clock, rounding error in the pulse compilation, or laser intensity instability, can lead to this loop not perfectly closing. As the motional mode heats up, the gates become more sensitive to this kind of noise, as explained in [181]. An error model very similar to this is used in [133]. We do not use a linear approximation for the exponential in

[181] to more accurately model higher motional mode excitations in our device.

In Honeywell's 4-qubit (8-ion) device [144], they report the average heating contribution of their shuttling operations as 2 quanta. This includes primitives such as split/merge and swap, as well as the simpler linear shuttles that the TILT architecture requires. Since split/merge operations and swap operations require complex and precise electrode voltages, these operations have significantly higher contributions to the heating rate than ion chain shuttling. As a result, we can fairly assume that the heating rate due to linear shuttles is lower than this number.

To scale this up to our experiments, we note that the primary consideration for heating is the softening of the common (center of mass) mode as the number of ions in the shuttled chain increases. The stopping force is still caused by the electrodes affecting the end of the chains, and is consequently constant [170]. However the mass of the chain increases, and as a result the noise in the system scales like a simple harmonic oscillator, with the frequency scaling by $\sqrt{n/F} \sim \sqrt{n}$.

4.4 Experimental Setup

4.4.1 Benchmarks

Table 4.2: List of benchmarks.

Application	Qubits	2Q Gates	Communication
ADDER	64	545	Short-distance gates
BV	64	64	Long-distance gates
QAOA	64	1260	Nearest-neighbor gates
RCS	64	560	Nearest-neighbor gates
QFT	64	4032	Long-distance gates
SQRT	78	1028	Long-distance gates

To evaluate TILT architectures against real applications, we choose multiple important quantum applications as our benchmarks (Table 4.2). The benchmarks are selected to have different program characteristics in order to show TILT properties for arbitrary quantum

circuits. According to communication patterns, there are three categories of applications. One consists of the applications with long-distance two-qubit gates, which require swap gates to become executable. The second category is for applications with short-distance two-qubit gates and hence do not necessarily need swap gates on TILT. The last category is nearest-neighbor two-qubit gates when the topology is in a 2D grid structure. This category will have short-distance two-qubit communication pattern when mapping on a linear topology.

The adder benchmark is based on the Cucarro adder [49]. Adders are important kernel functions in many quantum algorithms. Bernstein-Vazirani (BV) is a NISQ application commonly used to benchmark devices [20, 178, 55]. Quantum Approximate Optimization Algorithm (QAOA) [65] benchmark in our evaluation is a hardware-efficiency ansatz [129] for MaxCut problem. QAOA is a hybrid quantum-classical variational algorithm, and it is one of the most important quantum algorithms in the NISQ era. Random Circuit Sampling (RCS) has been proposed by Google to show quantum supremacy [28, 122, 8]. Quantum Fourier transform (QFT) [95] is an important function in many quantum algorithms (Shor's algorithm [156], phase estimation algorithm [44], and the algorithm for hidden subgroup problem [97]). This application uses Grover's search algorithm to find the square root number [95]. Grover's search algorithm is for database search, and it can achieve significant speedups compared with classical search algorithms [79, 95].

4.4.2 Simulation Parameters

We evaluate architectures with 60+ qubits and consider tape head sizes of 16 and 32. Using the noise model shown in Section 4.3.5, we estimate the fidelity of each gate after m tape moves.

All experiments are performed on a Ubuntu 16.04 system (Linux kernel 4.4-0-141-generic) with Intel Xeon Silver 4110 32-core CPU at 2.1 GHz and 128 GB of physical memory.

4.5 Evaluation

In our evaluation, we first show the impact of swap insertion. We then compare TILT systems to QCCD systems, and present the compilation and estimated execution times of each application.

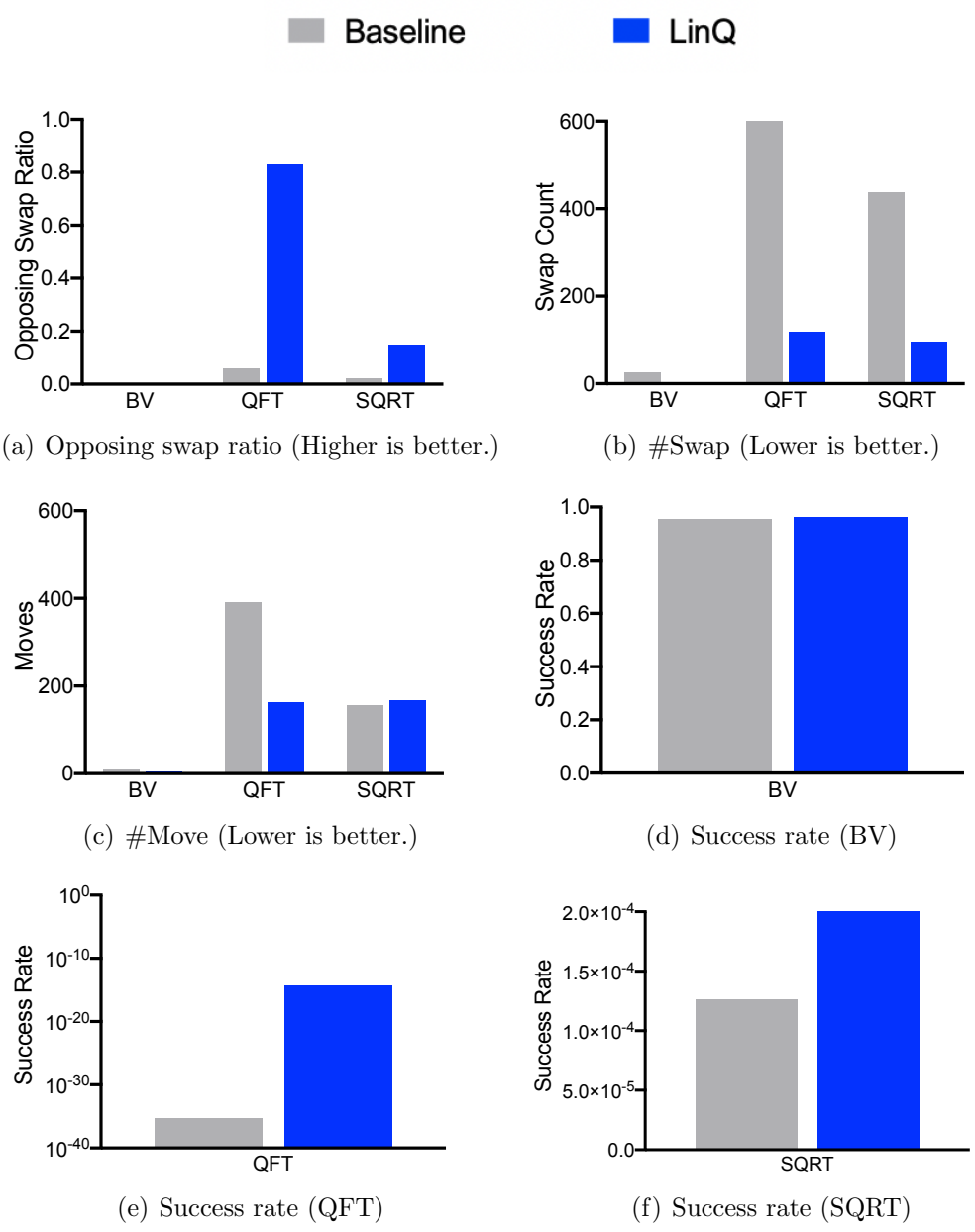
4.5.1 LinQ Swap Insertion Performance

In this section, we show the importance of swap insertion choices. We use only those applications with long-distance two-qubit gates (BV, QFT and SQRT) because no swap gates are needed for the other set of applications. We demonstrate the swap impact from two perspectives. First, we compare our LinQ swap insertion heuristic algorithm with a baseline method, which applies Qiskit StochasticSwap compilation tools to swap gate insertion for each unexecutable two-qubit gate. Second, we apply our swap insertion heuristic algorithm with a restricted swap distance to improve the tape movement scheduling. The results with our algorithm show increased circuit success rates.

We demonstrate the results with laser head size of 16. Figure 4.6(a) shows the opposing swap ratio, and Figure 4.6(b) shows swap counts with baseline and LinQ swap insertion. Our LinQ heuristic search reduces the total number of swaps and also increases the opposing swap ratio. For BV applications, however, LinQ does not create any opposing swaps because of the BV circuit structure [20, 178].

Figure 4.6(c) shows the number of tape moves (lower is better). Compared with the baseline, LinQ can synthesize circuits with fewer swaps and consequently fewer gates in total. As a result, fewer tape moves are required for the circuit execution. The success rate is related to the total number of gates and tape moves. Fewer gates and tape moves results in a higher success rate. Thus, LinQ can also achieve higher success rates, as shown in Figure 4.6(d) - 4.6(f).

Figure 4.7 demonstrates the effect of limiting swap length. Here we show the results of



(a) Opposing swap ratio (Higher is better.)

(b) #Swap (Lower is better.)

(c) #Move (Lower is better.)

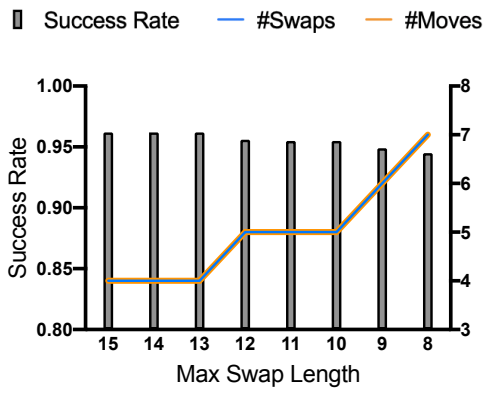
(d) Success rate (BV)

(e) Success rate (QFT)

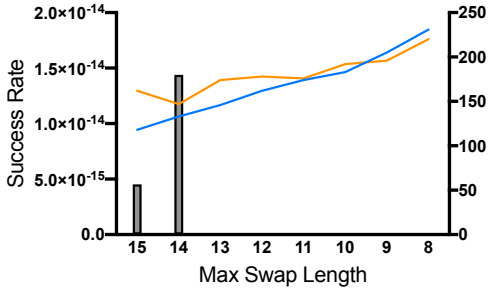
(f) Success rate (SQRT)

Figure 4.6: Comparing LinQ swap insertion with the baseline swap insertion.

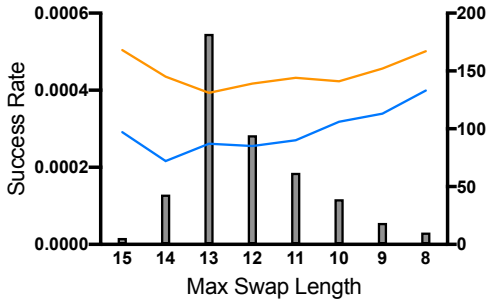
benchmarks on the devices with tape head size of 16. With LinQ swap insertion, while the swap is restricted to a shorter distance, the swap gate count could be slightly increased; but this restriction can potentially reduce the number of tape moves, leading to an improved success rate. Figure 4.7(a) shows the results of BV. The success rates are almost the same for the $MaxSwapLen$ of 15 through 13, because there is no difference in the number of swaps and tape moves. While the max swap length is limited to a lower number, the number of



(a) BV



(b) QFT



(c) SQRT

Figure 4.7: Success rates under different $MaxSwapLen$ restriction. For larger swap length, the circuit may need more tape moves. If the swap length is too short, however, the required number of tape moves increases as the total number of gates is increased. There is a swap length sweet spot depending on applications. For BV, the swap counts and move counts are the same so the lines are overlapped.

swap gates and moves are increased, and the success rate drops. Figure 4.7(b) shows the number of swaps and tape moves for QFT under different swap length restrictions. When the maximal swap length is 14, the number of tape moves is the lowest. Although the swap count is increased compared with $MaxSwapLen = 15$, the overall success rate is increased.

Therefore, we can get the highest success rate when the maximal swap length is shortened to 14. For SQRT, shown in Figure 4.7(c), when $MaxSwapLen = 13$, the swap count is slightly increased, but the tape moves can be significantly reduced. Hence, this application reaches the highest success rate at this configuration. For different applications, the best maximal swap length varies. We can iterate the LinQ procedure to find the best choice.

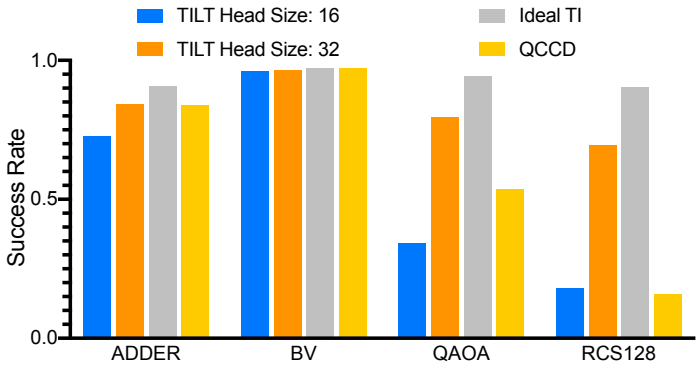
4.5.2 Architecture Comparison

In this section, we compare the TILT architecture with head sizes of 16 and 32 with ideal trapped-ion (Ideal TI) devices and the QCCD system presented in [133].

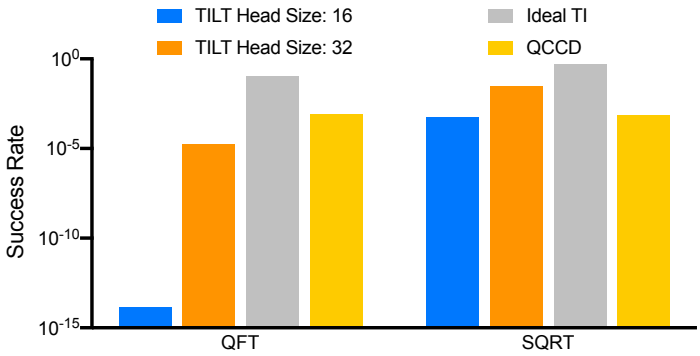
Ideal TI. An ideal trapped-ion device means that the device has enough laser controls for each qubit. Consequently, two-qubit gates can be performed on an arbitrary pair of qubits. As a result ion shuttling and swap gate insertion are unnecessary under our error model. With comparing to this ideal case, we can learn how close we are to the optimal solution.

QCCD. A recent study on QCCD is presented in [133]. In our evaluation, we apply the same noise model and topology evaluated in the study. We select the highest reported fidelity with AM gates for the comparison.

Figure 4.8 shows the results from our simulation. The success rate of TILT is related to the tape head size. For different tape head sizes, a larger tape head can cover a wider range of execution zone, reducing the number of swaps as well as tape moves and achieving a higher success rate. For ADDER and BV benchmarks, TILT has the same performance as QCCD. The reason is that the circuits are primarily operating on a few qubits. For QCCD, only a few shuttles have to be performed. Similarly for TILT, only a few tape moves are required to perform the circuit execution. However, for QAOA and RCS applications, the success rates of TILT are significantly higher than QCCD. As discussed in Section 4.2.3, frequent short-distance two-qubit gates will cause QCCD to perform split/merge and shuttling operations periodically if the involved qubits are not in the same trap, while TILT can possibly schedule those operations in one tape move. Since QAOA and RCS repeat the operations on the pair



(a) Success rates of applications (ADDER, BV, QAOA, and RCS).



(b) Success rates of applications (QFT and SQRT).

Figure 4.8: Success rates on different device configurations (higher is better).

of short-distance qubits, TILT outperforms QCCD in these results. For the circuits composed of short- and long-distance two-qubit gates (QFT and SQRT), TILT has lower success rate on QFT. This is as expected because the long-distance operations require multiple swaps and tape moves for TILT.

Our evaluation results suggest that TILT's success rate is better than that of QCCD for applications with a majority of the communication occurring within the width of a tape head.

4.5.3 Compilation Time and Execution Time

Table 4.3 shows the compilation results. The column labels t_{swap} and t_{move} are the compilation times for swap insertion and tape move scheduling for the benchmarks using our LinQ toolflow. The t_{swap} is short for each application. As discussed in Section 4.3.3, the

worst-case time complexity of searching for swap insertion is $O(NG)$. As a result, this heuristic search is scalable for larger applications. For tape move scheduling, as discussed in Section 4.3.4, the worst-case scheduling time is $O((N-L)DL)$. Since the circuit depth is increased when more swap gates are inserted, the scheduling time for QFT and SQRT is longer than for other applications. The number of swaps can be reduced by using a larger tape head, leading to lower circuit depth and a reduced t_{move}. The results and complexity analysis suggest that LinQ is applicable to larger circuits.

Table 4.3: LinQ compilation results.

Application	Head Size: 16					Head Size: 32				
	t_{swap}(s)	t_{move}(s)	#moves	dist(μm)	t_{exec}(s)	t_{swap}(s)	t_{move}(s)	#moves	dist(μm)	t_{exec}(s)
ADDER	0.002	6.214	10	104	2.967	0.002	3.248	5	68	3.252
BV	0.004	1.575	4	49	0.856	0.003	0.899	2	33	0.987
QAOA	0.011	2.750	18	232	1.564	0.011	1.101	4	72	1.357
RCS	0.001	3.148	65	992	1.704	0.001	0.680	11	214	0.856
QFT	9.206	55.617	162	2002	24.820	6.135	31.206	69	1276	33.876
SQRT	1.344	129.901	168	1816	46.554	1.075	55.234	76	1068	40.817

t_{swap}: compile time for swap insertion. t_{move}: compile time for tape move scheduling. **#moves**: total number of tape moves.
dist: total tape move distance. t_{exec}: program execution time.

To estimate the program execution time, we analyze the compiled circuit depth and tape move distance. We model the gate time as shown in Equation 4.3, and the shuttling rate (t_m) as $1\mu m/\mu s$ [37]. We obtain the program execution time as following,

$$t_{exe} = t_m \times dist + \sum_d t_d, \tag{4.5}$$

where t_d is the maximum gate time for the d-th depth. From our simulation, the applications can be finished within seconds. The results show that TILT machines are viable for running large applications, like our benchmarks, since trapped-ion qubits have long coherence times [166].

4.6 Trapped-Ion Scaling

Since ions are fundamentally identical and the same trapping parameters can handle a range of ion counts, scaling qubit count is not as significant a barrier as in other architectures. As we increase the number of ions, however, the heating cost of shuttling will also increase. This will lead to lower gate reliability, so there is an effective limitation on the number of ions in a single trap.

Various trapped-ion scaling techniques can be combined with TILT. In this section we discuss some additional techniques and technologies which could be used in conjunction with TILT to further scale trapped ion quantum computers.

Sympathetic Cooling. Gate reliability depends on the thermal energy in the chain. Sympathetic cooling is a technique to mitigate thermal heating, compensating for the increased heating rates expected in large-scale trapped-ion devices [132, 37]. A dual-species ion chain is composed of two different types of ions, one for storing information and which can be cooled by another laser-cooled ion during circuit execution without damaging the stored information. TILT architectures are compatible with sympathetic cooling techniques, which would reduce the heating due to shuttling and allow for longer circuits. With cooling ions, a few modifications would be made for LinQ toolflow to consider the tape scheduling under the head of cooling laser beams. However, the fundamental design goal is still the same, minimizing the number of tape moves.

QCCD Architectures. The QCCD architecture is a well-known design for constructing a large-scale quantum computer [102, 133]. QCCD systems have multiple small traps that are interconnected by segments and junctions. Before shuttling an ion from one trap to another, the ion must be split from the source chain. The split ion then is shuttled to the target trap and merged into the destination chain.

The operations necessary for individual ion shuttling have high costs in terms of shuttling complexity. Individual ion shuttling requires expensive split/merge and junction crossing maneuvers, which lead to more thermal energy entering the system. QCCD architectures

could be combined with TILT as a combined architecture where the TILT systems discussed in this work would be a primitive for constructing the QCCD. Trap capacities could become larger, which might be useful for applications in which the circuit naturally breaks up into larger densely-communicating chunks.

Such an architecture could combine the strengths of each system, allowing for circuits which have significant medium to long-range communication to live within a single trap, while also allowing other sections of the circuit which might not require such distant communication to occur within smaller higher-fidelity trapping zones. Heterogeneous systems like this, where different traps serve different purposes, would be a logical evolution of trapped ion quantum computers into a more modular framework similar to modern classical computers.

Photonic Interconnects. TILT can also be utilized in modular quantum computer architectures such as the modular universal scalable ion-trap quantum computer (MUSIQC) architecture proposed in [103, 130]. The element logic units (ELUs) in this architecture consist of arrays of ions, and one or more ions in each ELU are coupled to photonic quantum channels through photonic interconnects. With these interconnects, we can use TILT architectures as the fundamental building blocks of ELUs to achieve modular TILT architectures and further scale QC devices, similar to the QCCD discussion above.

4.7 Related Work

Several studies of qubit mapping and swap insertion problems have been carried out for superconducting systems. One common approach is to formulate the problems in mathematical form, such as integer linear programming, and then utilize software solvers to find the optimal solutions [11, 175, 121, 126, 180]. This method is guaranteed to provide an optimal solution. However, since the time to solution grows exponentially with the number of qubits and gates, so scaling this approach to large NISQ programs is infeasible. Another approach is to apply dynamic programming to find the optimal solutions [158, 94]. However, this method only works for circuits with 10 or fewer qubits since the time scaling is expo-

nential. To avoid long execution times, several researchers have chosen to apply heuristic search algorithms [94, 110, 187, 5, 149, 174, 105, 21]. Overall, most of the previous studies focus on compilation techniques for superconducting systems. In our work, we develop a scalable software toolflow to compile high-level quantum programs for TILT architecture, a novel trapped-ion system.

Previous studies have evaluated the performance of real devices with less than 20 qubits. In one case a fully connected 5-qubit trapped-ion system is compared with a 5-qubit superconducting transmon system [117]. In another study, several NISQ benchmarks are performed, comparing the trapped-ion system with superconducting systems [134]. These studies show that trapped-ion systems provide high program success rates because of the dense connectivity and higher gate fidelities compared with superconducting systems. In our work, we compare 64-qubit TILT systems to QCCD systems using our simulation tools and published results.

Several studies have proposed different strategies to scale trapped-ion quantum computers. Sargaran et al. proposed the SAQIP to apply reconfigurable optical interconnects for scalable trapped-ion systems [151]. Similarly, [103, 130] proposed the MUSIQC architecture that uses photonic interconnects to link modular elementary logic units. This architecture can scale to thousands of qubits and support fault-tolerant error correction. Lekitsch et al. [107] proposed a blueprint for scalable trapped-ion systems that use microwave-based quantum gates with on-chip control electronics to control an arbitrary number of qubits. While these studies focus on very large systems that are unlikely to be built in the next decade, TILT architectures provide a practical near-term implementation to scale trapped-ion systems.

A recent study focuses on 50–100 qubit scale modular QCCD-based trapped-ion devices [133]. They propose the use of simulation techniques to study the impact of trap sizes, topology, and gate implementations. In our work, we provide simulation techniques to study TILT architectures. Architectural simulations allow us to evaluate the performance of

different approaches of scaling trapped-ion systems before building them.

4.8 Conclusion

Trapped-ion technologies are a promising implementation for building practical quantum computers. Since all ions are identical, ions can be added to a long chain in order to scale the QC device. The TILT architecture is able to avoid issues with addressability as the number of ion addressed is held constant independent of chain length. Shuttling a small chain of ions has been demonstrated in [64], and larger TILT demonstrations are planned [34]. In this work, we show that TILT architectures offer a viable path toward QC devices approaching 60+ qubits. We present our optimizing compiler and simulator for TILT architectures. With a gate fidelity model derived from real experiments, our compiler performs qubit mapping, swap insertion, and tape move scheduling for NISQ applications within a few minutes. The results suggest that TILT can outperform QCCD in a range of NISQ applications. We also discuss using TILT as a building block to extend other scalable trapped-ion quantum computing proposals. Our evaluation of the TILT architecture offers insights for future architecture design.

CHAPTER 5

FULL-STATE QUANTUM CIRCUIT SIMULATION BY USING DATA COMPRESSION

Classical simulation of quantum circuits is crucial for better understanding the operations and behaviors of quantum computation. Such simulations allow researchers and developers to evaluate the complexity of new quantum algorithms and validate quantum devices. The path toward building Noisy Intermediate-Scale Quantum (NISQ) [145] machines such as IBM's 50-qubit quantum computer and Google's 72-qubit quantum computer [101] will require intermediate-scale quantum circuit simulators to calibrate and verify the hardware.

Unfortunately, today's practical full-simulation limit is 47 qubits (Table 5.1) [164]. The reason is that the number of quantum state amplitudes required for the full simulation increases exponentially with the number of qubits, making physical memory the limiting factor. Given n quantum bits (qubits), we need 2^n amplitudes to describe the quantum system [136]. In order to describe the amplitudes precisely, double-precision complex numbers are used to represent the state of the quantum systems. As a result, the size of the quantum state in the simulation is 2^{n+4} bytes. Although 49-qubit simulations will become possible in the near future with the arrival of exascale supercomputers [63], a gap still remains between the size of classical simulation and the size of NISQ machines.

Several simulation techniques related to Feynman paths [20] and tensor network contractions [29, 123, 142] have been proposed to trade time for space complexity. Different

Table 5.1: Examples of supercomputers, their total memory capacity, and the maximum number of qubits they can simulate for arbitrary circuits.

System	Memory (PB)	Max Qubits
Summit	2.8	47
Sierra	1.38	46
Sunway TaihuLight	1.31	46
Theta	0.8	45

approaches have different benefits and disadvantages. For Feynman paths method, both time and space complexity grow exponentially with the circuit depth, and thus this technique can simulate only shallow circuits. For tensor network simulation methods, since the time complexity grows exponentially with the underlying graph treewidth, these simulation techniques can only simulate the circuits with low treewidth. Some approaches calculate only a single amplitude or a partial state vector in order to be less restrictive on computational resources [41, 42, 142, 29]. As for quantum software development, several types of quantum applications require intermediate measurement [19, 74, 15, 31]. In addition, recent studies focus on quantum software debugging by inserting assertions in the middle of quantum programs [87]. Tensor network simulation techniques do not effectively support intermediate measurement and full-state assertion checking for software debugging. At the same time, NISQ machines are evolving toward supporting deeper circuits with error mitigation techniques [30, 99, 62, 150], which may make full-state simulations of high-depth circuits more important. When trying to verify quantum hardware and software by using a full-state simulator, every qubit counts in maximum simulation size. Every qubit closer to physical machine sizes means 2X more state space that can be evaluated, or, conversely, 2X smaller gap in state space between the simulation and the physical hardware.

To simulate general circuits with higher qubit count and depth, we propose quantum circuit simulation techniques that can reduce the memory requirement of the full simulation by compressing quantum state amplitudes at runtime. Specifically, we apply data compression techniques to the quantum state vector during the simulation. Since we aim to simulate intermediate-scale general quantum circuits, we have to achieve a data compression ratio as high as possible, because the compression ratio is the key to increasing the number of qubits in the simulation. Our approach uses lossless compression, lossy compression, and adaptive error bounds to reduce the memory requirement of the simulation. In general, lossy compression algorithms lead to significantly higher compression ratios than do lossless compressors, while introducing errors to a certain extent. To minimize the error propagation

and guarantee high-fidelity simulation results, we utilize both Zstandard lossless compressor [46] and an error-bounded tailored lossy compressor in our simulation framework.

Intuitively, one might be concerned that lossy compression would introduce correlated errors in our simulation output that would be very different from the kinds of errors that physical machines would experience. We will see, however, that our lossy compression is applied in an uncorrelated fashion. We shall also see that this has an added benefit of dramatically speeding up compression time.

Using our techniques, we are able to trade computation time and fidelity for memory space. We implement our techniques on Intel-QS, a full-state quantum circuit simulator developed by Intel [161]. Intel-QS is an MPI-based distributed high-performance quantum circuit simulator that can run on supercomputing systems. By orchestrating data compression techniques and full-state high-performance simulation techniques, our approach is capable of simulating intermediate-scale general quantum applications and hence obtain effective results of calibration, verification, and benchmarking for NISQ quantum machines.

Our approach integrates knowledge of quantum computation and data compression techniques to reduce the memory requirement of quantum circuit simulations such that our technique allows us to simulate a larger quantum system with the same memory capacity. We provide one more option in the set of tools to scale quantum circuit simulation. The main contributions of our work are as follows.

- We present a new technique to reduce memory requirements of full-state simulations of general quantum circuits by using data compression. Reducing memory requirements allows us to increase the number of qubits in our full-state simulations.

- We design a novel lossy compression method to optimize both compression ratios and compression speed for quantum circuit simulations. This lossy compression technique can be combined with several existing simulation techniques to further reduce the memory footprint.

- We implement our general quantum circuit simulation framework on the Theta supercomputer at Argonne National Laboratory (ANL).
- Our experimental results show that our approach reduces the memory requirement of simulating the 61-qubit Grover's search quantum circuit from 32 exabytes to 768 terabytes of memory. Based on the state-of-the-art simulation techniques, the results suggest that our technique can increase the simulation size by 2 to 16 qubits for general quantum applications with 0.976 simulation fidelity on average.

5.1 Background and Related Work

In this section, we provide a brief overview of the quantum computation and discuss related work on quantum circuit simulations. We then present the relevant background on compression techniques.

5.1.1 Principles of Quantum Computation

A qubit is a two-level quantum system, and the state $|\psi\rangle$ can be expressed as

$$|\psi\rangle = a_0 |0\rangle + a_1 |1\rangle , \tag{5.1}$$

where a_0 and a_1 are complex amplitudes and $|a_0|^2 + |a_1|^2 = 1$. $|0\rangle$, and $|1\rangle$ are two computational orthonormal basis states. The quantum state can also be represented as follows.

$$|\psi\rangle = a_0 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + a_1 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \tag{5.2}$$

More generally, the state of an n-qubit quantum system can be represented by using 2^n

amplitudes, as follows.

$$|\psi\rangle = a_{0...00}\,|0...00\rangle + a_{0...01}\,|0...01\rangle + ... + a_{1...11}\,|1...11\rangle \tag{5.3}$$

The squared magnitudes have to sum up to 1, that is,

$$\sum_i |a_i|^2 = 1. \tag{5.4}$$

In quantum computation, quantum gates are applied to the quantum system. All gates are represented in matrix form. General single-qubit gates and two-qubit controlled gates are known to be universal [61]. Applying a single-qubit gate U to the kth qubit can be represented by a unitary transformation

$$A = I^{\otimes n-k-1} \otimes U \otimes I^{\otimes k}, \tag{5.5}$$

where I is a 2×2 identity matrix and U is a 2×2 unitary matrix.

5.1.2 Quantum Circuit Simulation

Previous studies have provided various types of simulators [146]. Generally, in order to simulate a quantum circuit with n qubits and depth d, there are several simulation approaches [1, 20, 123]. Different simulation approaches have different purposes and benefits.

Schrödinger algorithm. This strategy maintains the full-state vector in memory and updates the state vector in every time step [53, 161]. Since the space grows exponentially with the number of qubits, the physical memory limits the simulation size. The time complexity is polynomial with the number of gates (or circuit depth), and hence this approach is capable of simulating arbitrary depth of circuits. This simulation approach can simulate the supremacy circuits of 45 qubits on the Cori II supercomputing system using 0.5 petabytes

[82], and Li et al. further optimize the simulator specifically for the 49-qubit supremacy circuits [111].

Feynman paths algorithm. This approach calculates the amplitude a_x for any n-bit string $x \in \{0,1\}^n$ by following all the paths from a final state to the initial state. For the Feynman paths algorithm, this approach requires $O(2^{dn})$ time to perform the simulation [142], and hence this simulation technique is suitable only for low-depth quantum circuits.

Tensor network contractions. This approach uses tensor networks to represent quantum circuits [123, 22, 127]. The time and space cost for contracting such tensor networks is exponential with the treewidth of the underlying graphs. Therefore, this approach is impractical to simulate large quantum circuits. Several studies have proposed to use tensor network simulation technique to simulate low-depth supremacy circuits [29, 142, 41, 169]. To trade the simulation fidelity for computational resources, the *approximate* simulation is proposed [122, 29]. The previous studies of approximate simulations target the overall circuit fidelity at 0.005 [122, 169], but if we want to use the simulation results to help calibrate and validate the real machines, we might need higher circuit fidelity. As for quantum software development, several types of quantum applications require intermediate measurement [19, 74, 15, 31]. In addition, recent studies propose to insert assertions in quantum programs for software debugging by checking the full-state distribution [87]. Tensor network simulation techniques do not effectively support intermediate measurement and full-state assertion checking.

Other strategies for simulating quantum circuits also exist. The *Gottesman-Knill theorem* [75] states that circuits consisting of only Clifford gates can be efficiently simulated in polynomial time, hence there are simulation techniques for the circuits with a restricted gate library [72, 75, 2, 32]. In [188], the decision diagram is used to simulate circuits that consist of Clifford+T gates. This approach exploits redundancies in a quantum state to gain more

compact representations. Approximate simulation techniques also have been proposed for circuits using only restricted gates [186, 189, 32].

5.1.3 Data Compression Techniques

With the vast volumes of data being produced by extreme-scale scientific research and applications, various data compression techniques have been developed for years. Basically, scientific researchers mainly adopt two types of compressors: lossless compressors or error-bounded lossy compressors. Lossless compressors usually adopt both variable-length encoding algorithms (such as Huffman encoding [89] and arithmetic encoding [177]) and dictionary coders such as LZ77/78 [184]. In most cases, however, lossless compressors such as Gzip [56], Zstd [46], and Blosc [25] cannot effectively compress scientific data because the ending mantissa bits of floating-point values are random such that it is hard to find exactly the same patterns in the data stream. Some studies [57] show that lossless compressors always suffer from low compression ratios (around 2:1 in most cases), which is far less than enough for today's extreme-scale high-performance computing (HPC) applications. Accordingly, error-bounded lossy compression has been widely treated as the best solution to such a big scientific data issue, because it not only can significantly reduce the data size but also can control the data distortion according to the user's requirements.

Error-bounded lossy compressors may have distinct designs and implementations, so selecting the most appropriate compression technique is critical to our research. All existing error-bounded lossy compressors can be categorized into two models: data-prediction based and domain-transform based, which are described below.

- *Data-prediction-based compression model.* This model tries to predict each data point as accurately as possible based on its neighborhood in spatial or temporal dimension and then shrinks the data size by some coding algorithm such as data quantization [165] and bit-plane truncation. A typical example compressor is SZ [112], which involves four compression steps: (1) data prediction, (2) linear-scaling quantization, (3) entropy-

encoding, and (4) lossless compression. The errors are introduced and controlled at step (2). Other examples include ISABELA [106] and FPZIP [116].

- *Domain-transform-based compression model.* This model needs to transform all the original data values to another nonorthogonal coefficient domain for decorrelation and then shrink the data size by applying some optimized coding algorithms such as embedded coding [115]. A typical example compressor is ZFP [115], which performs the classic texture compression by leveraging three techniques in each 4^d block (where d is the number of dimensions): (1) exponent alignment, (2) (non)orthogonal block transform, and (3) embedded coding. The compression errors (or distortion of data) are introduced and controlled only at step (3). The other two techniques are VAPOR [45] and Sasaki et al.'s compressor [152], which both adopt the Wavelet transform in the domain transform step.

All the existing state-of-the-art error-bounded lossy compressors are designed or assessed mainly for visualization, so they are not optimized for the requirement of data fidelity and compression quality in the context of quantum computing simulation. In this sense, we first characterize the effectiveness of the existing state-of-the-art lossy compressors on quantum circuit simulation results and then exploit a fairly efficient compression method beyond the existing lossy compressors. In our study, we investigate two types of error controls, pointwise absolute error bound and pointwise relative error bound, because they have been supported by the existing state-of-the-art lossy compressors well.

- Absolute Error Bound (denoted by e). With this type of error control, the compression errors (defined as the difference between the original data value and its corresponding decompressed value) of all data points must be strictly limited in the required bound; in other words, d_i' must be in $[d_i - e, d_i + e]$, where d_i and d_i' refer to an original data value and the corresponding decompressed value, respectively.

- Relative Error Bound (denoted by ϵ). With this type of error control, the data distor-

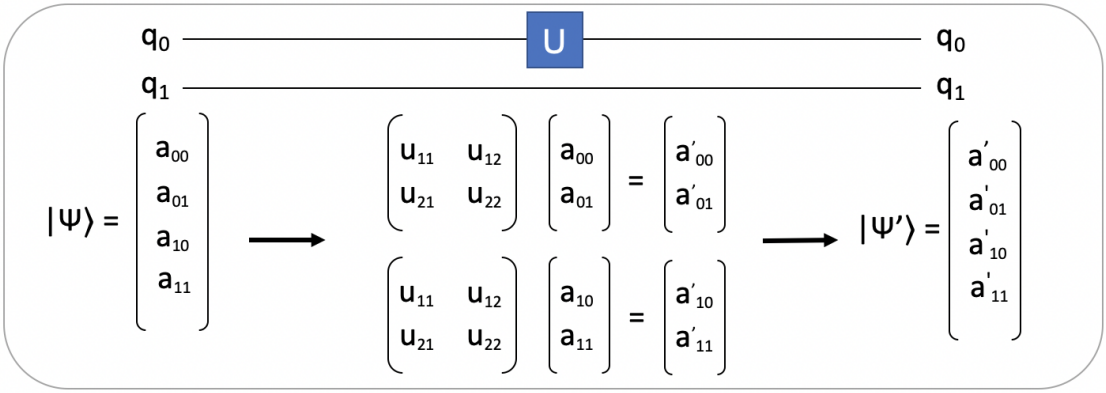


Figure 5.1: Example of two-qubit quantum state with single-qubit gate operations

tion must respect the following inequality for each data point: $|d_i - d_i'| \leq \epsilon d_i$. Obviously, the smaller the original value is, the smaller the compression error it will get. The relative error bound is particularly useful to applications requiring multiple error controls depending on the data values.

5.2 Simulation Design

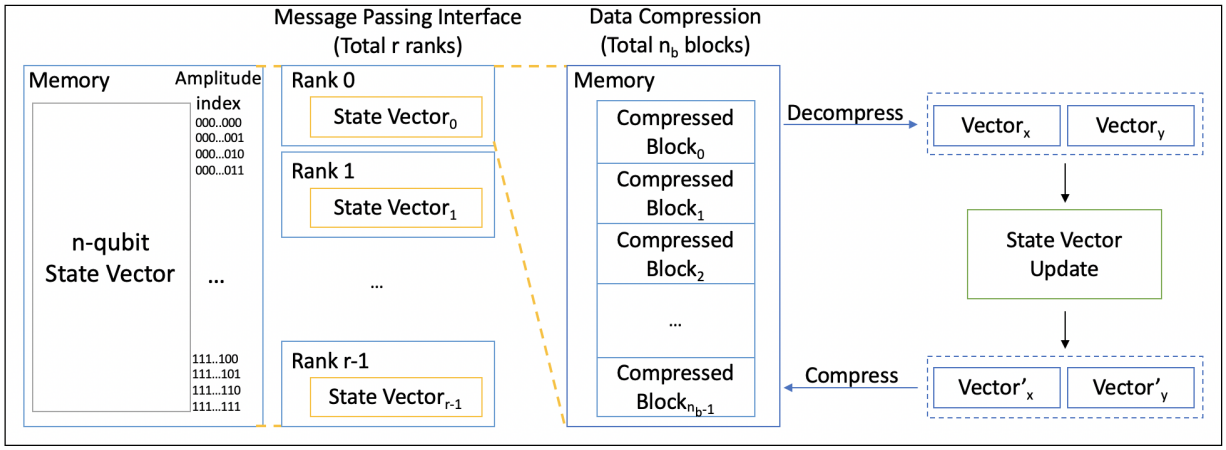


Figure 5.2: Simulation overview

We aim to simulate general quantum circuits with high fidelity. We integrate our compression techniques into the quantum circuit simulation such that the simulation scale can be increased with the same memory capacity. Our technique allows the Schrödinger-style simulation to trade the computation time and simulation accuracy for memory space by

applying lossy compression techniques to state vectors. The lower compression error bound gives us the higher simulation fidelity, but the higher compression error bound will give us a higher compression ratio so that we can simulate the quantum circuits with larger numbers of qubits.

5.2.1 Overview of Our Simulation Flow

To demonstrate the practicality of our design, we integrate our compression techniques into Intel-QS [161], a distributed quantum circuit simulator on a classical computer.

As mentioned in Section 5.1, applying a single-qubit gate to the kth qubit can be represented by a unitary transformation $A = I^{\otimes n-k-1} \otimes U \otimes I^{\otimes k}$. In the simulation, however, we do not need to build the entire unitary matrix A to perform the gate operation. Figure 5.1 shows an example of applying a single-qubit gate to a two-qubit system. Applying a gate U to the first qubit corresponds to applying the 2×2 unitary matrix to every pair of amplitudes, whose subscript indices have 0 and 1 in the first bit and all remaining bits are the same. In the same way, performing a single-qubit gate to the second qubit is to apply the unitary to every pair of amplitudes whose subscript indices differ in the second bit. More extensively, applying a single-qubit gate to the k-th qubit of an n-qubit quantum system is to apply the unitary to every pair of amplitudes whose subscript indices have 0 and 1 in the k-th bit, while all other bits remain the same.

$$\begin{bmatrix} a'_{*\ldots*0_k*\ldots*} \\ a'_{*\ldots*1_k*\ldots*} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{bmatrix} \begin{bmatrix} a_{*\ldots*0_k*\ldots*} \\ a_{*\ldots*1_k*\ldots*} \end{bmatrix} \tag{5.6}$$

As for a generalized two-qubit controlled gate, the unitary is applied to a target qubit t if the control qubit c is set to $|1\rangle$; otherwise tth is unmodified.

$$\begin{bmatrix} a'_{*1_c\ldots*0_t*\ldots*} \\ a'_{*1_c\ldots*1_t*\ldots*} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{bmatrix} \begin{bmatrix} a_{*1_c\ldots*0_t*\ldots*} \\ a_{*1_c\ldots*1_t*\ldots*} \end{bmatrix} \tag{5.7}$$

Figure 5.2 shows an overview of our simulation design. The Message Passing Interface (MPI) [77] is used to execute the simulation in parallel.

Assuming we simulate n-qubit systems and have r ranks in total, the state vector is divided equally on r ranks. To reduce the memory requirement, we further divide the partial state vector into n_b blocks on each rank. Each block is stored in compressed format on the memory. To complete a gate operation, we need to apply matrix multiplication to the pair of amplitudes whose subindices are 0 and 1 at the target qubit position, so at most two blocks are decompressed, $Vector_x$ and $Vector_y$, and then update the state vectors. After all the amplitudes in $Vector_x$ and $Vector_y$ are updated, we compress the state vectors and move to the next two blocks. Once all the blocks have been updated, a gate operation is completed.

In this way, the total number of bytes required for the simulation of a rank is as follows:

$$nbBytes = \sum_i sizeof(CB_i) + 2(\frac{2^{n+4}}{r \times n_b}), \tag{5.8}$$

where CB_i is the ith compressed block.

5.2.2 MCDRAM Memory Configuration

Multichannel DRAM (MCDRAM) is a high-bandwidth, low-capacity memory, packaged with the Intel Xeon Phi processor [162]. Because several repeated compression and decompression operations are involved in the simulation, we can achieve the performance improvement by utilizing MCDRAM. Every time the block is decompressed, we decompress the state vectors to MCDRAM. To allocate memory in MCDRAM, we use *mkl_malloc* function to acquire memory space. This memory allocation strategy directly improves the performance of compression, gate operation, and decompression. To achieve this, we set the machine to *equal* mode, 50% cache and 50% flat.

5.2.3 Integration Details

We build our compressor as a C library and add it into the Intel-QS building process[1]. In the initialization of the simulation process, we create the state vectors in blocks, and compress them as compressed blocks. During the simulation, if the process needs to update the state vector, it calls our compressor library to decompress the block to the pre-allocated MCDRAM. After the state vector update, the block is compressed, and the process moves to the next block.

Since we allow decompression of only two blocks for each rank at the same time, we must select the corresponding blocks to be decompressed. For n-qubit systems, assuming we have r ranks and each block contains b amplitudes, the amplitude index string can be divided into three segments (Figure 5.3). When we execute a single-qubit gate computation on the target qubit position q, we find the corresponding blocks according to the segment q belongs to.

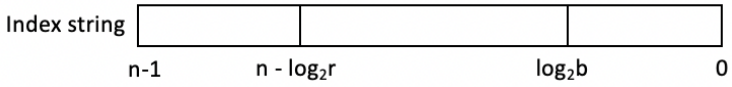


Figure 5.3: Amplitude index segments

- $q < log_2 b$: Both amplitudes are in the same block.
- $n - log_2 r \leq q \leq log_2 b$: Both amplitudes are in the same rank but with different blocks. We can use the number q to find the corresponding blocks in the rank.
- $q > n - log_2 r$: The pair of amplitudes are in the different ranks. The blocks have to be exchanged between different ranks.

Two-qubit gate operations are similar to single-qubit gate operations, but we modify the amplitudes only when the control qubit is set to $|1\rangle$. According to the position of the control qubit c, we also have three cases to determine whether the amplitudes should be modified:

1. More integration details and the source code can be found at https://github.com/ryanxw/compressed_qcsim.

- $c < log_2 b$: If cth bit is 0, the amplitude is left unmodified.
- $n - log_2 r \leq c \leq log_2 b$: If cth bit is 0, the whole block is left unmodified.
- $c > n - log_2 r$: If cth bit is 0, the whole rank is left unmodified.

5.2.4 Compressed Block Cache

For most of the quantum circuits, the amplitudes may share the same value [188]. By exploiting redundancies in the quantum state, we can reduce the computation time significantly by constructing a compressed block cache.

Figure 5.4: Compressed block cache line

Figure 5.4 shows the contents of a compressed block cache line. The first element (OP) is the gate operation and the target qubit position. The second and third elements (CB_1, CB_2) are the compressed blocks before the gate operation. The rest of the elements (CB'_1, CB'_2) are the compressed blocks after the gate operation. Each rank maintains a memory space for the compressed block cache with 64 cache lines. When a gate operation is executed, our computation procedure first checks whether the pattern (OP, CB_1, CB_2) is in the cache. If there is a cache hit, then the computation is done by directly returning the blocks (CB'_1, CB'_2), and thus the performance is improved by reducing the compression, computation, and decompression time.

The cache replacement policy is least recently used. If there is no redundancy in the quantum state, the cache hit rate will be 0, and this will introduce the cache miss penalty in our simulation. Thus, our simulator will disable the compressed block cache if the cache hit rate is always zero.

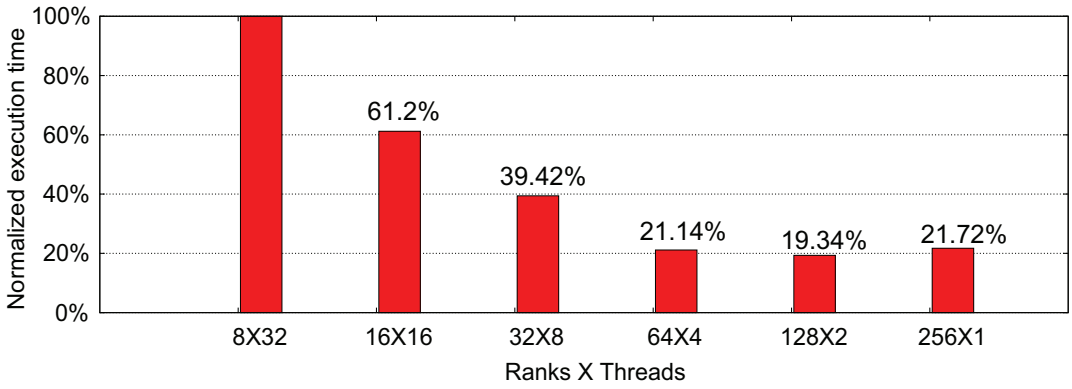


Figure 5.5: Normalized execution time for running 35-qubit random circuit simulations.

5.2.5 Simulation Checkpoint

In general, most supercomputing systems have a 24-hour wall-time limit. This runtime constraint puts a circuit depth limit on the simulation. However, one can save the compressed blocks before terminating the job and then resume the task by loading the compressed blocks in the next job submission.

5.2.6 MPI Configuration

To understand the performance under different MPI rank configuration, we run the 35-qubit random circuit simulation with various ranks per node (Figure 5.5). On the KNL nodes, each node has 64 cores and 256 threads. We found that the setting of 128 ranks per node gives the best performance.

5.2.7 Variable Error Bound Compression

To keep the simulation accuracy, we use the lossless compression Zstd[46] in the beginning of the simulation when the compression ratio is still high enough to fit all compressed blocks in the memory space. During the simulation, the quantum state becomes more and more complicated, and hence the lossless compression ratio is lower. When the compression ratio is too low to fit all compressed blocks in the memory, our simulation will use lossy compression

to compress the state vectors. To control the error, we use a pointwise relative error bound to compress the state vector. This compression mode guarantees that the decompressed data point $|D'|$ must be in the range of $(|D(1-\delta)|, |D|)$, where D is the original data and δ is the error bound. In this work, we have five different levels of error bounds: 1E-5, 1E-4, 1E-3, 1E-2, and 1E-1. Whenever the compression ratio is not enough, the error bound is relaxed to the next level (larger error). We give a detailed discussion about our lossy compression technique in the next section.

5.2.8 Lower Bounds on Simulation Accuracy

Since lossy compression is used in our simulations, information is lost with every lossy compression, causing a decrease in the simulation's overall accuracy. The accuracy of our simulation can be quantified by the state fidelity, a measure of the similarity of two quantum states [136].

The fidelity takes values between 0 and 1, with higher fidelity indicating greater similarity between the two states. A fidelity of 1 would indicate that the two quantum states are the same. The pure state fidelity between the ideal output state, ψ_{ideal}, and the output state from our simulation, ψ_{sim}, can be described by the following simplified equation [136].

$$F(ideal, sim) = |\langle \psi_{ideal} | \psi_{sim} \rangle| \tag{5.9}$$

Since the errors are bounded in our simulation, the fidelity can be estimated by propagating the maximum error bounds at each gate to calculate the maximum decrease in fidelity (from the ideal fidelity of 1) that comes from each lossy compression and then finding their combined impact on the overall maximum decrease in fidelity. Suppose that a gate has a percentage error of δ. Then if a complex amplitude a_i in the ideal state vector is represented as $a + bi$, the corresponding amplitude after our lossy compression in the simulation can be represented by $a' + b'i$, where $|a'| \geq |a(1-\delta)|$ and $|b'| \geq |b(1-\delta)|$. Thus, we see that the

state fidelity can be calculated as follows:

$$F(ideal, sim) = |\langle\psi_{ideal}|\psi_{sim}\rangle| \geq \sum_{i=0}^{2^n-1} a_i^2(1-\delta) = 1-\delta. \tag{5.10}$$

So we see that the minimum fidelity drops by a factor of $(1-\delta)$ after applying a lossy compression with a maximum percentage error bound of δ. Suppose that for the ith gate, the error bound is δ_i on the lossy compression. Then the lower bound on the fidelity after applying this compression will drop to $(1-\delta_i)$ times the lower bound on the fidelity calculated before lossy compression on this gate. Combining the contributions of all the gates in the simulation allows us to calculate the lower bound on the simulation fidelity as

$$F(ideal, sim) \geq \prod_i (1-\delta_i). \tag{5.11}$$

If all the δ_i were 0, the simulation fidelity would be 1, which makes sense because our simulation would then be identical to the simulation without lossy compression against which we are calculating the fidelity. As mentioned before, in our simulation with lossy compression, the error bounds δ_i can be set to 0 (lossless), 1E-5, 1E-4, 1E-3, 1E-2, and 1E-1. Figure 5.6 shows how fidelities change with the number of gates when different error levels are applied.

5.3 Adaptive Compression Optimized for Quantum Circuit Simulation

In this section, we propose a novel error-bounded compression method that can significantly control memory footprint for full-state quantum circuit simulations that were unreachable before.

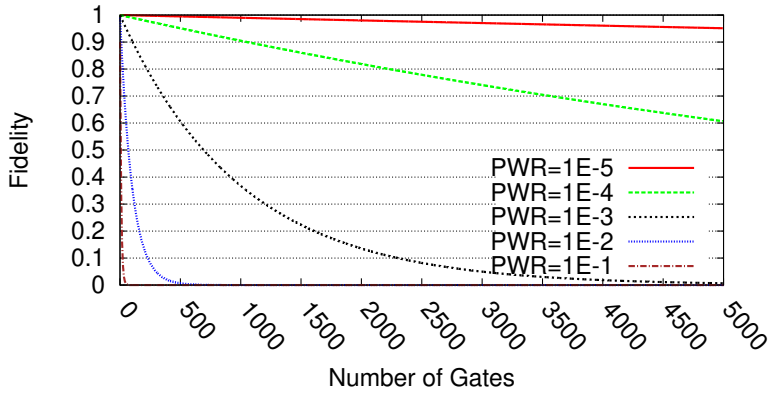


Figure 5.6: Minimum bounds for fidelity with an increasing number of gates at different error levels

5.3.1 Assessment of Existing Error-Bounded Lossy Compressors

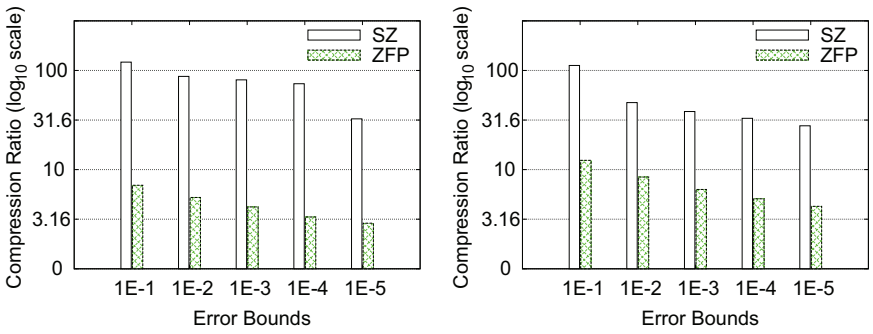
First, we explore the best solution from among the existing compressors for the quantum circuit simulations. We choose SZ [58, 185], FPZIP [116], and ZFP [115] in our exploration because they have been confirmed as the best error-bounded compressors on many scientific datasets [58, 185, 119].

We perform the assessment using two well-known quantum circuits: a quantum approximate optimization algorithm (QAOA) [65] and the random circuit proposed by Google to show quantum supremacy [28]. In this analysis, we run 36 qubits for both circuits and denote them as qaoa_36 and sup_36, respectively.

As discussed in Section 5.1.3, there are two types of error bounds, both of which have been widely used by scientific applications, so we evaluate the compression ratios for quantum circuit simulation data based on both types of errors. Since each rank involves hundreds or thousands of data blocks each having different value ranges, we perform the compression based on the absolute error bound in the regard of value range in our evaluation, without loss of generality. That is, we set the absolute error bound to a fixed percentage of the value range in each block. For instance, 1E-2 means 1% of the value range in the following figures.

Figure 5.7 presents the compression ratios of SZ and ZFP based on different absolute error bounds. FPZIP is missing in this figure because it does not support an absolute error bound.

We can see that SZ always leads to one or two orders of magnitude higher compression ratios than ZFP does at every error bound. For instance, on qaoa_36, SZ can lead the compression ratios up to about 100:1, while ZFP's compression ratios are always less than 10:1. For the dataset sup_36, the compression ratios of SZ and ZFP are about 28∼126 and 4.25∼12.6, respectively.

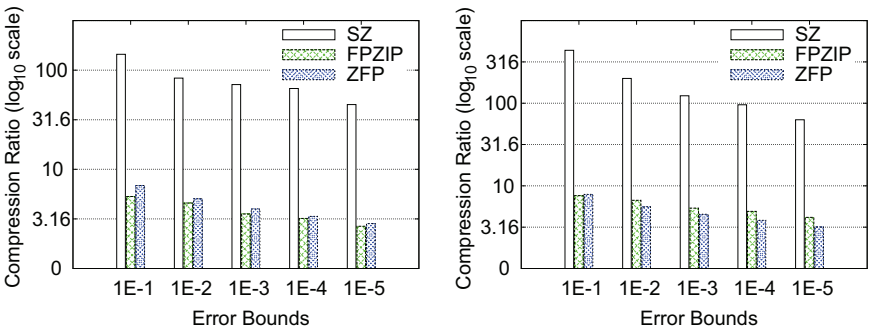


(a) qaoa_36 (b) sup_36

Figure 5.7: Compression ratio of SZ vs. ZFP (absolute error)

For the pointwise relative error bound, for ZFP, we transform the original data to the logarithm domain and then compress the transformed data with absolute error bounds, for fairness of the comparison. Such a log-preprocessing-based compression has been validated as the best way to do the pointwise relative-error-bounded compression [113]. As for FPZIP, it provides a so-called precision number (=4∼64) to control the pointwise relative error bound. The larger the precision number is, the lower the pointwise relative error bound obtained. We set the precisions to 16, 18, 22, 24, and 28 for FPZIP in our experiments because they correspond to the pointwise relative error bounds of 1E-1, 1E-2, 1E-3, 1E-4, and 1E-5 approximately. Figure 5.8 clearly shows that SZ always leads to much higher compression ratios than do the other two compressors with the same pointwise relative error bounds.

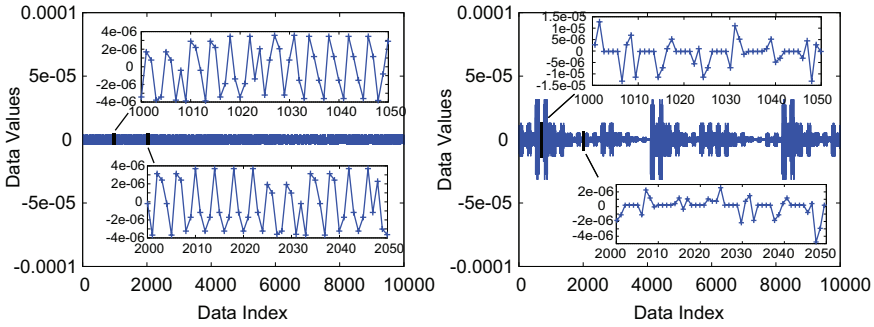
Based on our analysis, we conclude that SZ is superior to the other two compressors. The key reason is as follows. For ZFP, it substantially replies on the high smoothness of data, especially when the error bounds are set to a relatively low value. However, the quantum



(a) qaoa_36 (b) sup_36

Figure 5.8: Compression ratio of SZ,FPZIP,ZFP (relative error)

simulation data are not smooth at all, as illustrated in Figure 5.9, such that the domain-transform in ZFP would totally lose its effectiveness, leading to poor compression ratios. The key difference between FPZIP and SZ is that the former adopts a totally different encoding method, unlike SZ adopting linear-scaling quantization + Huffman encoding + Zstd. In what follows, we treat SZ as the baseline and propose a new lossy compression method that is more effective on the quantum circuit simulation data.



(a) qaoa_36 (b) sup_36

Figure 5.9: Illustration of Value changes of quantum circuit simulation data: the data exhibit a high spikiness such that existing lossy compressors cannot work effectively.

5.3.2 Optimizing the Compression Strategy

In our work, we adopt a hybrid, adaptive compression pipeline to control the memory footprint while keeping the high fidelity of the simulation results and relatively low time overhead. Specifically, we observe that at the early stage of the simulation, a large majority of the data

are zero. In this situation, the lossless compressor Zstd can also get a satisfactory compression ratio, without any loss of data fidelity. Although SZ leads to much higher compression ratios in this situation, it may introduce data distortion, causing degraded fidelity unexpectedly. As the simulation goes on, however, the simulation data will become more and more complicated, such that Zstd will suffer from low compression ratios. In this situation, the error-bounded lossy compressor is helpful in controlling the memory footprint while keeping a high fidelity of simulation results. Thus, we adopt both Zstd and error-bounded lossy compression during each entire simulation. More details are in Section 5.2.7.

Developing a fast, effective error-bounded lossy compressor is critical to the overall simulation results (including fidelity and simulation time). In particular, we need to use a pointwise relative error bound to do the compression, as discussed in Section 5.2. To this end, we explored a battery of novel compression strategies to optimize both the compression ratios and compression speed for quantum circuit simulations, as described below.

Solution A (SZ 2.1): This solution is the state-of-the-art error-bounded lossy compressor - SZ [58, 165, 185]. In the context of quantum circuit simulations, the data can be treated only as a 1D array. In SZ 2.0, the pointwise relative-error-bounded compression involves the following steps: (1) logarithm data transform; (2) absolute-error-bounded compression on log-transformed data (Lorenzo prediction [90] + quantization [165]); (3) Huffman encoding; and (4) Zstd lossless compression. Since log-transform is expensive, the SZ development team developed SZ 2.1 leveraging a table lookup method to accelerate the compression significantly [185], without degrading the compression ratios. However, the compression/decompression speed still cannot meet the expected level for quantum circuit simulations (to be shown later), which motivates us to further explore a new, faster compression method instead.

Solution B (SZ 2.1 with complex type supported): Quantum state amplitudes are stored in the form of complex data type, in which the real numbers and imaginary num-

bers are stored alternatively, such that the prediction accuracy would be degraded to a certain extent, leading to limited compression ratios. Accordingly, Solution B improves the prediction accuracy by performing the prediction on the real numbers and imaginary numbers, respectively. In addition, we set the maximum number of quantization bins to 16,384 (unlike the default setting of 65,536 in SZ 2.1), which can significantly improve the compression/decompression rate (to be shown later).

Solution C (XOR leading-zero data reduction + bit-plane truncation + Zstd): Solution B can improve the compression ratios in some cases, but its compression speed is still lower than expected such that the total simulation would slow significantly compared with the original compression-free execution. To reduce the compression/decompression time significantly, we developed Solution C, a simple yet efficient compression pipeline that is particularly suitable for quantum circuit simulation. This method involves three key steps. For each data point, it first leverages *XOR leading-zero data reduction method* [38, 58], which uses a two-bit code to record the number of exactly the same bytes between the current value and its preceding value. Then the algorithm truncates the insignificant bit-planes based on the required relative error bound ϵ. Specifically, the significant number of bits can be calculated as follows.

$$Sig\_Bit\_Count = Bit\_Count(Sign\&Exp) - EXP(\epsilon), \tag{5.12}$$

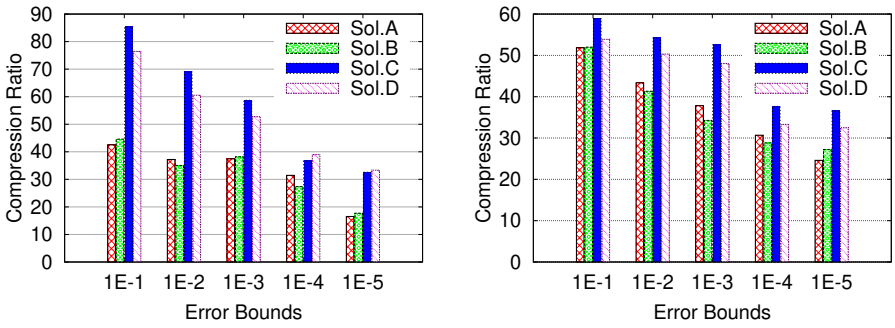
where $EXP(\epsilon)$ refers to the exponent of the relative error bound ϵ (e.g., $EXP(0.01)=-7$) and $Bit\_count(Sign\&Exp)$ is the total number of bits used to represent sign and exponent in the IEEE 754 format (e.g., it is equal to 12 for double precision). We then adopt Zstd lossless compression to shrink the data significantly.

Solution D (Reshuffle + Solution C): Since the simulation data are stored in the complex

data type (i.e., real number and imaginary number alternatively), one plausible idea is to reorganize the data into real numbers and imaginary numbers separately before compressing the data. Such a reshuffle step may help improve compression ratio especially when the real numbers and imaginary numbers are located in different non-overlapped value ranges. The reason is that in Solution C, the last step Zstd involves a dictionary-matching stage (LZ77 [184]) leveraging potential repeated patterns in the data streams and the reshuffling step that separates the real numbers and imaginary numbers may improve the pattern matching to a certain extent. Accordingly, we developed Solution D, which might have higher compression ratios with a slightly lower compression/decompression rate.

We evaluate all the four solutions by doing the compression and decompression with the two simulation datasets qaoa_36 and sup_36. Figure 5.10 presents the compression ratios of the four solutions. We can see that the classic compressor SZ 2.1, either supporting complex type (Solution B) or not (Solution A), suffers from about 30%∼50% lower compression ratios than do Solutions C and D. The likely reason is that the simulation data exhibit spiky changes (as illustrated in Figure 5.9) such that the SZ compressor always suffers from low prediction accuracy. Besides, we note that the solution C may lead to slightly higher compression ratios than does the solution D in some cases, which also makes sense as explained as follows. Note that the only difference between these two solutions is the extra reshuffle step in the solution D. This step might affect the compression ratio only because of the LZ77 stage in the last step Zstd of the two solutions, as analyzed previously. With this in mind, the compression ratio actually may not change significantly in between since the pattern matching efficiency for the two solutions could be very similar. On the one hand, Zstd improved LZ77 by adopting a pretty large window size. On the other, the real numbers and imaginary numbers are of the similar value ranges (as shown in Figure 5.9), such that the reshuffling step may not induce more regular data. In this sense, the solution C and D can be deemed having comparative compression ratios on the QC simulation datasets.

We present the compression and decompression rate in Figure 5.11 (single-core perfor-



(a) qaoa_36 (b) sup_36

Figure 5.10: Compression ratio of 4 solutions (relative error)

mance). This figure clearly shows that Solutions B, C, and D have much higher compression rates and decompression rates than the classic SZ 2.1 (Solution A) does. The key reason Solution B is faster than Solution A is that it predicts the data in terms of the complex data type, which may get higher prediction accuracy, thus leading to faster encoding thereafter. Moreover, we set a lower maximum number of quantization bins in Solution B such that it keeps a relatively high compression rate in the case with low relative error bounds such as 1E-5. The reason the Solutions C and D run much faster than Solutions A and B do is that they totally remove the three costly steps—prediction, quantization, and Huffman encoding—in the compression. We can also observe that Solution C is slightly faster than Solution D because of the extra reshuffle step in Solution D.

We present in Figure 5.12 the distribution of the maximum compression error (pointwise relative errors) per data block for all four compression solutions on qaoa_36 and sup_36, respectively. Similar to the previous evaluations, the experimental dataset comes from one rank involving hundreds of data blocks (each with 1,048,576 complex-type data points, bringing a total of 16 MB per block).

From Figure 5.12 we observe that all four solutions respect the compression errors well in terms of different pointwise relative error bounds (1E-1∼1E-5). We note that the error distribution curves of Solutions C and D overlap, which is explained as follows. In fact, they both avoid the data prediction and quantization step such that the compression errors have

nothing to do with the order of the data points. The only difference between them is the extra reshuffle step in Solution D. Thus they have exactly the same compression errors in each block.

Moreover, we note that Solutions C and D exhibit much lower compression errors than do Solutions A and B in general. The reason is as follows. As illustrated in Figure 5.13 (a), Solutions A and B both adopt the data prediction and linear-scaling quantization [165]. Specifically, they predict each data point and then approximate its value by the error-bound-quantized distance [165] between the true raw value and the predicted value. Note that the compression errors are determined by the difference between the decompressed values and the true values (as shown in Figure 5.13 (a)). Hence, if the error bound is relatively small, the quantization bin size would be small too, then the true values would be located at a rather random postion in a quantization bin because of the likely large distance between the predicted value and true value, thus leading to a uniform distribution. By comparison, Solutions C and D calculate the significant bit-planes based on user-required pointwise relative errors, and each bit-plane spans a certain value range. As illustrated in Figure 5.13 (b) (with a single-precision value 3.9921875 as an example), truncating different bit planes will lead to discrete decompressed values and relative errors. Suppose the relative error bound is set to 0.01, then we need to keep 15 leading bits and the decompressed value would be 3.96875 with a relative error of 0.005871, which is actually lower than the error bound 0.01. That is, the compression errors of the solutions C and D are generally somewhat lower than the desired error bound.

Such overpreservation of error can be observed more clearly by plotting the distribution of normalized compression errors compared with the error bounds for Solution C, as presented in Figure 5.14. We plot the cumulative distribution function (CDF) of the normalized pointwise relative errors compared with the corresponding error bounds for one random block of data, because too many data points are involved in the whole simulation and other blocks actually exhibit similar error distributions. In the figure we can observe that (1)

all compression errors are indeed confined within the required relative error bound; (2) the compression errors follow a uniform distribution; and (3) most of the compression errors are actually much lower than the required error bound, bringing an extra benefit to the control of data distortion in the simulation.

Moreover, the solution C leads to non-correlated compression errors (point-wise relative errors) for the quantum circuit simulation datasets. The reason is that the relative errors are determined by the high-order bit-values in the truncated insignificant bit-planes. Since the quantum circuit simulation data exhibit a rather high randomness (as presented in Figure 5.9), the truncation errors are supposed to be fairly random as well. We confirm this point by calculating the lag-1 autocorrelation coefficients of the compression errors. The autocorrelation value is always in [-1,1]; the closer to zero, the higher level of non-autocorrelation is. Our evaluation shows that the autocorrelation value often ranges in [-1E-4,1E-4] if a large majority of original raw data are non-zeros in the dataset, testifying the high non-correlation feature of the solution C.

Based on this analysis, we can conclude that Solution C is a fairly good tradeoff, which can obtain a high compression ratio, low compression time and decompression time, and low compression errors with non-correlation feature. Accordingly, Solution C is our final error-bounded lossy compressor to be used in our experiments.

5.4 Evaluation

5.4.1 Experimental Setup

For multinode evaluation, we performed our simulation on the Theta supercomputer at Argonne National Laboratory. Theta consists of 4,392 nodes, each node containing a 64-core Intel®Xeon Phi™ processor 7230 with 16 gigabytes of high-bandwidth in-package memory (MCDRAM) and 192 GB of DDR4 RAM [141]. The bandwidth of the MCDRAM is 400GB/s, and the average latency is 154 ns [9]. On Theta, there are wall-time limits for each

jobs with different numbers of nodes. The wall-time limits are 3, 6, and 24 hours for jobs of 128 nodes, 256 nodes, and 1,024 nodes, respectively. For jobs running exceeding wall-time limits, we use our checkpoint design, described in Section 5.2.5, to complete the simulation.

For single-node experiments, we run our simulation on a single 64-core Intel®Xeon Phi™ processor 7210 (KNL) in the Joint Laboratory for System Evaluation (JLSE) at Argonne. There is no wall-time limit on the system.

In this evaluation, we set 128 ranks per node. As described in Section 5.2, the state vector on each rank is divided into multiple blocks, each containing 1,048,576 amplitudes, which means each block is 16 MB. Since each rank decompresses at most 2 blocks on MCDRAM at the same time, we need to allocate $128 \times 2 \times 16MB = 4GB$ of memory on MCDRAM. Thus, we configure the memory mode by using *--attrs mcdram=equal* to operate 50% MCDRAM as flat memory and the other 50% as cache [96].

5.4.2 Scalability

We evaluate the scaling behavior by using a basic program that applies a Hadamard gate on each qubit. Figure 5.15 shows the normalized execution time for running different sizes of simulations on a single node. The scaling of our simulation technique for 51-qubit quantum circuit running on different numbers of nodes is depicted in Figure 5.16. For simulations of real applications that requires more time in compression, decompression, and computation, the scaling is expected to be better.

5.4.3 Benchmarks

To show the results against real applications, we choose multiple important quantum applications as our benchmarks. The benchmarks are selected to have different program characteristics in order to show that our simulation could work with arbitrary quantum circuits. Our approach performs well for both shallow-circuit and deep-circuit applications. The initial state is $|0\rangle^{\otimes n}$. The programs we use for our benchmarking include the following.

- Grover: Grover's search algorithm is for database search, and it leads to significant speedups compared with classical search algorithms [79, 95]. Our benchmark uses Grover's search algorithm to find the square root number [95], and thus the oracle consists of X and Toffoli gates.

- Random circuit sampling: The circuit is proposed by Google to show the quantum supremacy, and we follow the rules to construct the random circuits [28]. Since our approach is not optimized for the circuits, we do not plan to run the random circuits with many layers. Thus, the circuit depth is 11 in our experiments.

- QAOA: The quantum approximate optimization algorithm is a hybrid quantum-classical variational algorithm. Our benchmark uses QAOA to solve MAXCUT on a random 4 regular graph problem [65]. QAOA is an important circuit because it is one of the most promising quantum algorithms in the NISQ era [145].

- QFT: This is the quantum circuit for quantum Fourier transform [95], which is an important function in many quantum algorithms (Shor's algorithm [156], phase estimation algorithm [44], and the algorithm for hidden subgroup problem [97]), and this is a deep circuit. We randomly apply X gate to the initial state as the input for the QFT in our experiments.

Table 5.2: Experimental results. The first row shows the memory requirements of the simulations without our techniques. The ratios of our system memory sizes to the required memory sizes are presented in the fourth row. The fifth row provides the simulation time and the breakdown. The minimum compression ratio during the simulation is shown in the last row.

Benchmark	Grover			Random Circuit Sampling				QAOA			QFT
Number of Qubits (Memory Requirement)	61 (32 EB)	59 (8 EB)	47 (2 PB)	5×9 (512 TB)	6×7 (64 TB)	6×6 (1 TB)	7×5 (512 GB)	45 (512TB)	43 (128 TB)	42 (64 TB)	36 (1 TB)
Number of Gates	314	310	305	227	261	165	208	394	344	336	3258
Number of Nodes	4096	4096	128	1024	128	1	1	1024	256	128	1
Total System Memory (Sys Mem / Req.)	768 TB (0.002%)	768 TB (0.009%)	24 TB (1.17%)	192 TB (37.5%)	24 TB (37.5%)	192 GB (18.75%)	192 GB (37.5%)	192TB (37.5%)	48 TB (37.5%)	24 TB (37.5%)	192 GB (18.75%)
Total Time (Hour)	8.14	3.48	0.49	4.87	8.64	7.96	6.23	13.34	5.83	8.65	78.98
Compression Time	1.87%	4.59%	2.04%	55.79%	40.26%	59.10%	58.57%	50.66%	44.97%	41.02%	57.86%
Decompression Time	1.87%	3.73%	4.08%	31.47%	22.19%	33.78%	30.59%	26.46%	27.64%	25.52%	37.68%
Communication Time	32.7%	20.98%	36.73%	0.12%	0.57%	0.02%	0.03%	3.03%	0.22%	0.23%	2.56%
Computation Time	63.47%	70.70%	57.15%	12.60%	36.97%	7.08%	10.8%	19.84%	27.16%	33.22%	1.9%
Time per Gate (Sec)	93.34	40.49	5.78	64.69	119.22	173.65	107.86	121.91	61.02	92.64	87.27
Simulation Fidelity	0.996	0.996	1	0.987	0.993	0.933	0.985	0.895	0.999	0.999	0.962
Compression Ratio	7.39×10^4	8.26×10^4	1.06×10^4	6.03	9.40	8.16	10.05	5.38	4.85	9.25	21.34

5.4.4 Experimental Results

We present our main results in Table 5.2. We run benchmarks with the total system memory capacities much less than the theoretical memory requirements to manifest the strength of our approach.

Our first benchmark application is Grover's search algorithm with 47, 59 and 61 qubits. Previously, 61-qubit simulations of general quantum applications were thought to be impossible because of infeasible memory requirements. Using our methods, the state vectors can be compressed with high compression ratios because the amplitude values of the state vectors of this application are similar and regular, and hence our approach can use only 768 TB to successfully perform the simulation of 61-qubit Grover's search algorithm, which theoretically requires 32 EB to execute the full-state simulation. We also use our technique to complete the 47-qubit Grover's search algorithm simulation using 24 TB instead of 2 PB, the memory requirement without our technique. Next, we present the simulation of Google's quantum supremacy random circuits. Since our method exploits non-uniformity and structure in quantum computations, it does not work as well on random circuits. If we run many layers of the circuits, we have to drop the simulation fidelity to meet the available memory capacity. Thus, we present the simulation results of depth of 11 random circuits. Although our simulation technique is not designed specifically for the supremacy circuits, our approach can simulate the circuit with 42 qubits by using 48 TB, and simulate the circuit with 45 qubits by using 192 TB. The next benchmark is QAOA. Since QAOA is an important quantum application, it is critical to the simulation of QAOA circuits with limited memory capacity. In fact, we can get higher compression ratios and still get successful results because QAOA is robust to low-fidelity. Finally, the results with QFT circuits show that our techniques are effective for simulating high-depth circuits and compress the state vectors with more than 21X compression ratios, with a fidelity of 0.962.

For the quantum applications we choose, we show that our technique can simulate the circuits with high fidelity. Among the selected benchmarks, random circuits use more en-

tanglement than others. Since our techniques compress the state vector, more entanglement leads to less compressible vectors. Thus, our technique does better when there is less entanglement.

5.4.5 Discussion

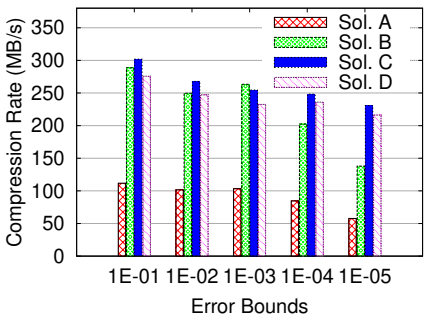
We successfully increase the maximum simulation size from 45 qubits to 61 qubits for the Grover's search circuit simulation on the Argonne Theta supercomputer using less than 0.8 PB. As for simulations of the other quantum applications, we have compression ratios from 4.85X to 21.34X. The results provide evidence that we are able to simulate those applications with 47 to 49 qubits on Theta. In other words, for simulations of general circuits, our approach can increase the simulation size by 2 to 16 qubits. In this work, we seek to achieve high-fidelity simulation results for general circuits. For different purposes that do not require high-simulation fidelity, the compression ratios and simulation size would be further increased.

We use the compression ratios to estimate our approach on the Summit supercomputer [172] at Oak Ridge National Laboratory (ORNL). The expected maximum simulation size for general circuits is 63 qubits. In 2021, we will have the exascale supercomputing system Aurora [63] at Argonne. The estimated maximum simulation size will be increased from 48 qubits to 64 qubits. Our compression techniques can combine with other simulation techniques [42, 82, 111, 186, 189] to scale quantum circuit simulation.

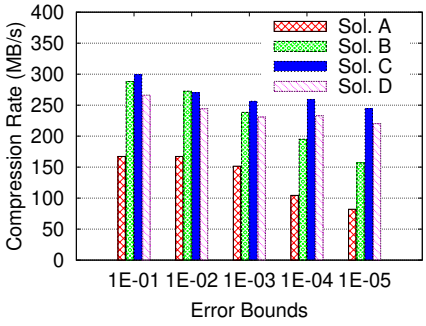
We trade time and slight fidelity for memory space. This trade-off makes the classical simulation of several quantum circuits possible, while the simulation time grows linearly with the number of gates. The time complexity is polynomial with circuit depth.

5.5 Conclusion

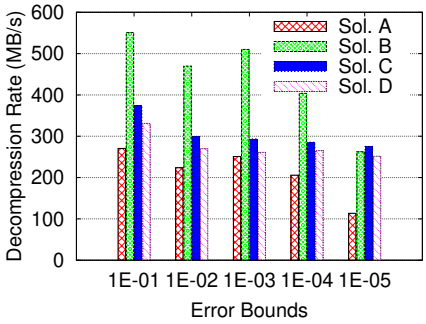
In summary, we have described our simulation techniques to reduce the memory requirement of full-state quantum circuit simulations by using data compression techniques. Our method provides a new option in the set of simulation tools to scale quantum circuit simulations. We propose a novel lossy compression technique to optimize compression ratios and compression speed for quantum circuit simulations. Using our approach, the memory requirement of simulating the 61-qubit Grover's search algorithm is reduced from 32 exabytes to 768 terabytes of memory on the Argonne Theta supercomputer using 4,096 nodes. We also present experimental results of random circuits, QAOA, and QFT simulations to show that our technique can achieve general circuit simulations. The compression ratio results further suggest that our technique can increase the simulation size by 2 to 16 qubits for general quantum applications. Our simulator provides a platform for quantum software debugging and quantum hardware validation.



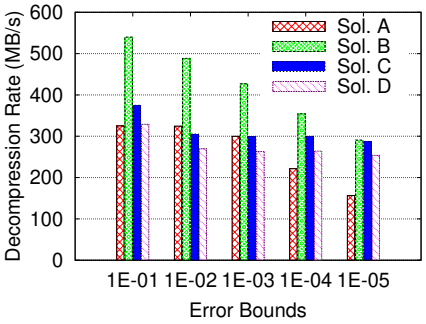
(a) qaoa_36 (Cmpr)



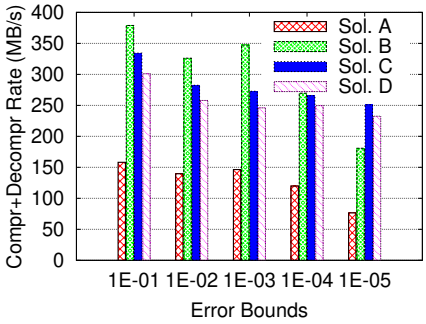
(b) sup_36 (Cmpr)



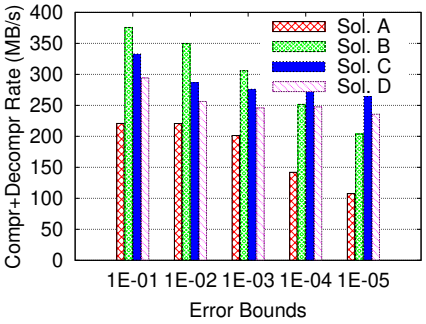
(c) qaoa_36 (Decmpr)



(d) sup_36 (Decmpr)

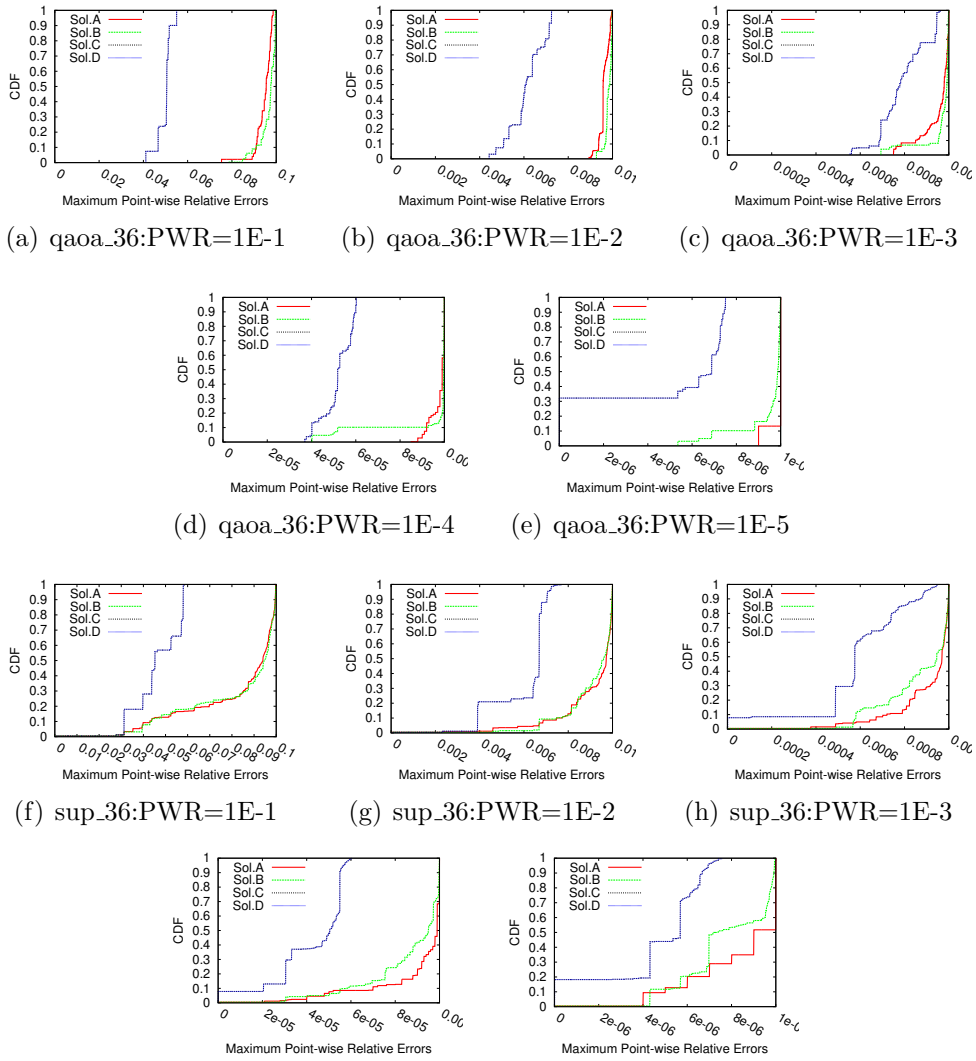


(e) qaoa_36 (Cmpr+Decmpr)



(f) sup_36 (Cmpr+Decmpr)

Figure 5.11: Compress/decompression rates (relative error)



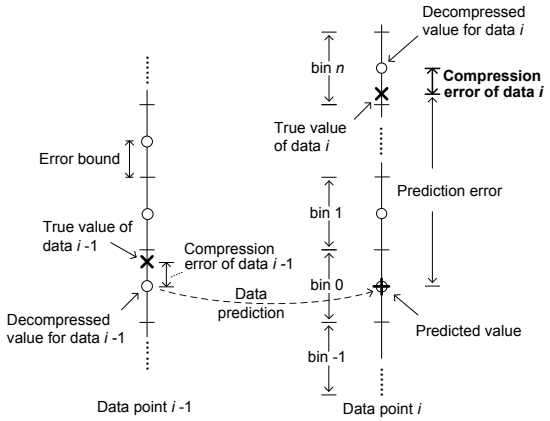
(a) qaoa_36:PWR=1E-1 (b) qaoa_36:PWR=1E-2 (c) qaoa_36:PWR=1E-3

(d) qaoa_36:PWR=1E-4 (e) qaoa_36:PWR=1E-5

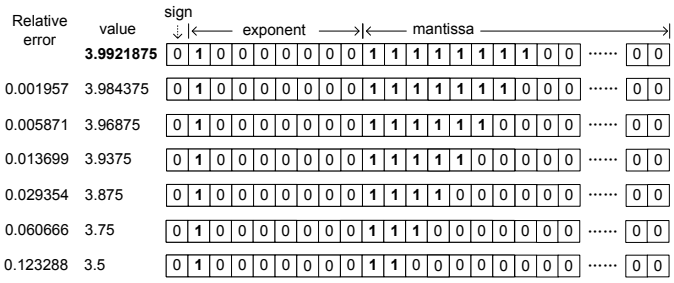
(f) sup_36:PWR=1E-1 (g) sup_36:PWR=1E-2 (h) sup_36:PWR=1E-3

(i) sup_36:PWR=1E-4 (j) sup_36:PWR=1E-5

Figure 5.12: Distribution of maximum relative errors on compression of qaoa_36 and sup_36 (Solution C and D overlap)

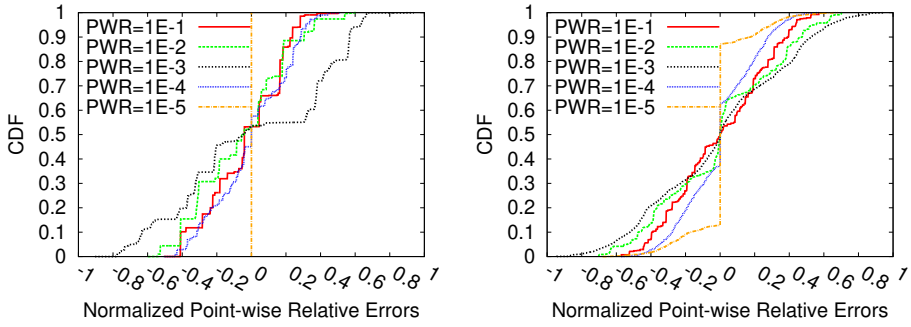


(a) Prediction and quantization in solution A/B



(b) Discrete relative errors when truncating bits in solution C/D

Figure 5.13: Illustrating why Solution C/D leads to lower compression errors than Solution A/B



(a) qaoa_36 (b) sup_36

Figure 5.14: Distribution of normalized compression errors (Solution C)

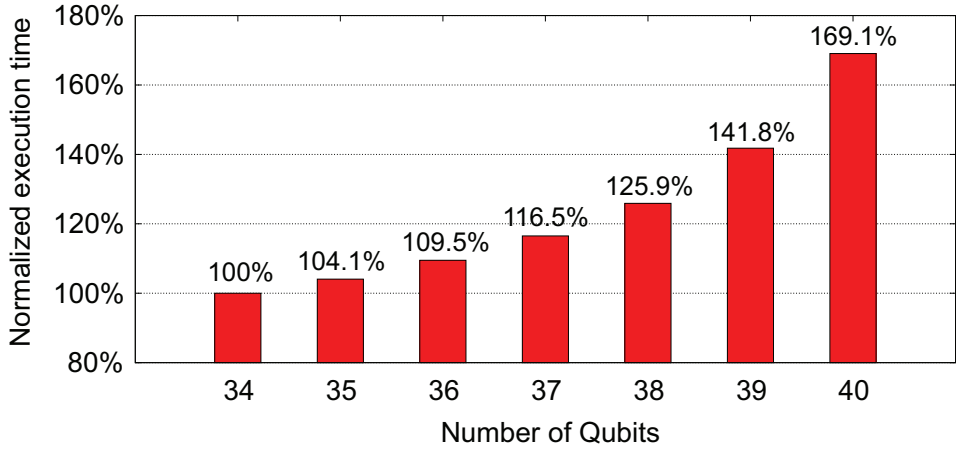


Figure 5.15: Normalized execution time for running various sizes of simulations on a single node.

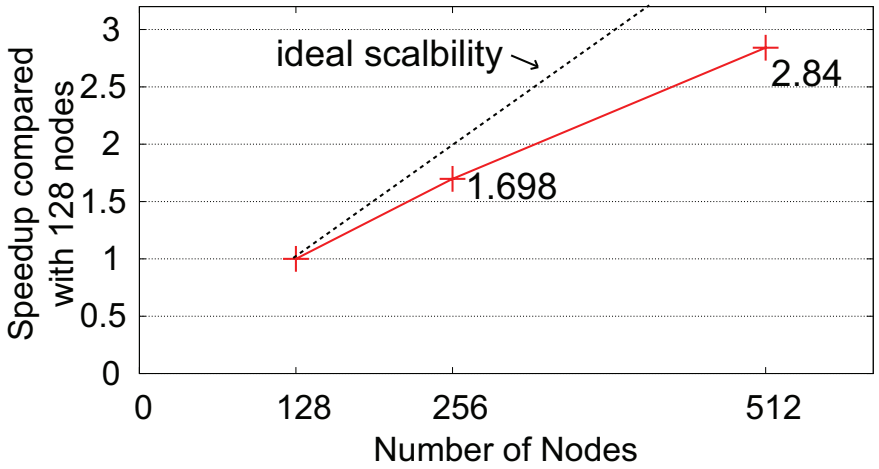


Figure 5.16: Scaling behavior of applying Hadamard gates on a 51-qubit system on Theta.

CHAPTER 6

FUTURE WORK

In this chapter, I discuss some future directions to extend this line of research.

6.1 Quantum Circuit Optimization using Synthesis

Even though Artisan already achieves successful results in optimizing different sizes of circuits, we discuss a few directions that would improve the development of quantum computing.

6.1.1 Approximate Synthesis

The objective of optimization using synthesis is to reduce the CNOT count. In some cases, it may be desirable to make approximations to reduce the number of CNOTs at the expense of how closely the final circuit approximates the desired unitary. An intuitive way to achieve this is to relax a synthesis threshold. The synthesis tool that we used is capable of doing this, but even when loosening the threshold as far as $10^{-}4$, we found no reduction. The limiting factor on approximation in this case is that the heuristic, process infidelity, is not flexible in a way needed to find good approximations. In order to effectively trade accuracy for CNOT count, a threshold-controllable synthesis tool is necessary. Integrating a different synthesis tool or heuristic that is designed for approximate synthesis may further improve the optimization.

6.1.2 Pulse-Level Optimization

Since the device-level control of a quantum computer is operated via analog pulses, recent pulse optimization studies aim to generate shorter pulses [69, 154, 70, 71, 43]. Pulse optimization can also be integrated into our Artisan technique. The current synthesis output is a sequence of quantum gates. To integrate pulse optimization into this work, we can synthesize the block into a sequence of pulses.

6.1.3 Artisan in Parallel

In this work, we perform the quantum synthesis in serial. However, since we partition a circuit into a sequence of circuit blocks, each block is independent for the synthesis process. As a result, the synthesis of blocks can be executed in parallel. Running the entire Artisan on a supercomputing system can reduce the overall compilation time significantly.

6.2 TILT Architecture Scaling

Since ions are fundamentally identical and the same trapping parameters can handle a range of ion counts, scaling qubit count is not as significant a barrier as in other architectures. As we increase the number of ions, however, the heating cost of shuttling will also increase. This will lead to lower gate reliability, so there is an effective limitation on the number of ions in a single trap.

Various trapped-ion scaling techniques can be combined with TILT. In this section we discuss some additional techniques and technologies which could be used in conjunction with TILT to further scale trapped ion quantum computers.

6.2.1 Sympathetic Cooling

Gate reliability depends on the thermal energy in the chain. Sympathetic cooling is a technique to mitigate thermal heating, compensating for the increased heating rates expected in large-scale trapped-ion devices [132, 37]. A dual-species ion chain is composed of two different types of ions, one for storing information and which can be cooled by another laser-cooled ion during circuit execution without damaging the stored information. TILT architectures are compatible with sympathetic cooling techniques, which would reduce the heating due to shuttling and allow for longer circuits. With cooling ions, a few modifications would be made for LinQ toolflow to consider the tape scheduling under the head of cooling laser beams. However, the fundamental design goal is still the same, minimizing the number

of tape moves.

6.2.2 QCCD Architectures

The QCCD architecture is a well-known design for constructing a large-scale quantum computer [102, 133]. QCCD systems have multiple small traps that are interconnected by segments and junctions. Before shuttling an ion from one trap to another, the ion must be split from the source chain. The split ion then is shuttled to the target trap and merged into the destination chain.

The operations necessary for individual ion shuttling have high costs in terms of shuttling complexity. Individual ion shuttling requires expensive split/merge and junction crossing maneuvers, which lead to more thermal energy entering the system. QCCD architectures could be combined with TILT as a combined architecture where the TILT systems discussed in this work would be a primitive for constructing the QCCD. Trap capacities could become larger, which might be useful for applications in which the circuit naturally breaks up into larger densely-communicating chunks.

Such an architecture could combine the strengths of each system, allowing for circuits which have significant medium to long-range communication to live within a single trap, while also allowing other sections of the circuit which might not require such distant communication to occur within smaller higher-fidelity trapping zones. Heterogeneous systems like this, where different traps serve different purposes, would be a logical evolution of trapped ion quantum computers into a more modular framework similar to modern classical computers.

6.2.3 Photonic Interconnects

TILT can also be utilized in modular quantum computer architectures such as the modular universal scalable ion-trap quantum computer (MUSIQC) architecture proposed in [103, 130]. The element logic units (ELUs) in this architecture consist of arrays of ions, and one or more ions in each ELU are coupled to photonic quantum channels through photonic interconnects.

With these interconnects, we can use TILT architectures as the fundamental building blocks of ELUs to achieve modular TILT architectures and further scale QC devices, similar to the QCCD discussion above.Since ions are fundamentally identical and the same trapping parameters can handle a range of ion counts, scaling qubit count is not as significant a barrier as in other architectures. As we increase the number of ions, however, the heating cost of shuttling will also increase. This will lead to lower gate reliability, so there is an effective limitation on the number of ions in a single trap.

Various trapped-ion scaling techniques can be combined with TILT. In this section we discussed some additional techniques and technologies which could be used in conjunction with TILT to further scale trapped ion quantum computers.

6.3 Quantum Circuit Simulation by using Lossy Compression

Our approach performs full-state simulation of general quantum circuits using data compression techniques. This method allows the Schrödinger-style simulation to trade time and simulation fidelity for memory space to increase the simulation size. Our compression techniques can combine with other simulation techniques [42, 82, 111, 186, 189].

6.3.1 Noise Simulation

By using our lossy compression, we can compress state vectors and reduce memory footprints significantly compared with the existing techniques [161, 82]. The compression errors are not correlated to the data, and hence the errors might be used to further simulate noise on real devices. The modern noise simulations add errors to perfect simulations. However, we could further adapt our lossy compression errors to noise models and then build a simulation which models noise naturally. In addition, we plan to implement our simulator on GPU-based supercomputing systems to reduce the compression and decompression time.

CHAPTER 7

CONCLUSION

People are interested in quantum computing because quantum computing shows the capability of solving the problems that used to be intractable in classical computing. By leveraging the quantum mechanical principles of superposition, interference, and entanglement, quantum algorithms have the potential to revolutionize areas such as machine learning, chemistry, optimization, and cryptography. However, However, in reality, quantum systems are extremely sensitive to environmental effects. Qubits and gates are not perfect. There is a gap between quantum software and currently available hardware. In the current NISQ era, the machine is large enough to support interesting experiments, but too small for known algorithms with exponential speedup, and too small for error correction code. My research focuses on using compiler techniques and architecture design to improve the circuit fidelity of quantum computing, and hence reducing the gap.

In the first part of this thesis, I present our scalable optimization by using synthesis, which is an automated compilation framework, Artisan. It partitions the circuit into blocks, and re-generates each optimized block by using synthesis, and re-composes the circuit by stitching all the blocks together. This approach to circuit optimization offers a role for quantum synthesis algorithms in large-scale quantum computing scenarios. It can reduce the gate count, and improve circuit fidelity. I also demonstrated the fidelity improvement against real devices and simulators.

In the second part of this thesis, I discuss the architecture specific optimization for scalable TILT architecture. I show that TILT architectures offer a viable path toward QC devices approaching 60+ qubits. I present our optimizing compiler and simulator for TILT architectures. With a gate fidelity model derived from real experiments, our compiler performs qubit mapping, swap insertion, and tape move scheduling for NISQ applications within a few minutes. The results suggest that TILT can outperform QCCD in a range of NISQ applications.

In the last part, I show our work on scaling full statevector simulation by using data compression. Our method provides a new option in the set of simulation tools to scale quantum circuit simulations. I propose a novel lossy compression technique to optimize compression ratios and compression speed for quantum circuit simulations. Using this approach, the memory requirement of simulating the 61-qubit Grover's search algorithm is reduced from 32 exabytes to 768 terabytes of memory on the Argonne Theta supercomputer using 4,096 nodes. The results show that our method can increase the simulation size.

Hopefully this thesis inspires more creative ideas, and help researchers to build up the community and involve more people to join our journey to achieve practical and scalable quantum computing.

APPENDIX A

CURRICULUM VITAE

Xin-Chuan (Ryan) Wu

Website: xinchuanwu.com
Email: xinchuanwu@gmail.com

Education

University of Chicago — Chicago, IL
Ph.D. Candidate in Computer Science, Advisor: Prof. Frederic T. Chong — 2016–Current
- Research Topic: Quantum Computing
- Thesis: Design, Optimization, and Simulation of Scalable Quantum Computing Systems

National Taiwan University — Taipei, Taiwan
M.S. in Computer Science and Information Engineer, Advisor: Prof. Chia-Lin Yang — June 2009
- Thesis: System-level Power Estimation for Digital Signal Processor

National Chiao Tung University — HsinChu, Taiwan
B.S. in Computer Science — June 2007
- Research Topic: Asynchronous Circuit Chip Design

Experience

Lawrence Berkeley National Laboratory — Berkeley, CA
Grad Student Summer Intern — Jun–Sep 2020
- Quantum Circuit Optimization Using Synthesis

Argonne National Laboratory — Lemont, IL
Research Aide — Jun–Sep 2019
- ILP-Based Scheduling for Linear-Tape Model Trapped-Ion Quantum Computers
- Achieving Higher Fidelity on a Trapped-Ion Linear-Tape Quantum Computing Architecture

Argonne National Laboratory — Lemont, IL
Research Aide — Jun–Sep 2018
- Full State Quantum Circuit Simulation by Using Data Compression

ASUS Computer International — Fremont, CA
Senior Software Engineer/R&D Specialist — Oct 2013–Jun 2016
- Led a team to develop Linux kernel driver for Android systems such as IoT and mobile devices.

ASUS Computer Inc. — Taipei, Taiwan
Software Engineer — Oct 2009–Sep 2013
- Linux kernel drive development.

Publications

[1] **X.-C. Wu**, D. M. Debroy, Y. Ding, J. M. Baker, Y. Alexeev, K. R. Brown, and F. T. Chong, "TILT: Achieving Higher Fidelity on a Trapped-Ion Linear-Tape Quantum Computing Architecture", in *Proceedings of 27th IEEE Symposium on High Performance Computer Architecture (HPCA)*, Feb. 2021.

[2] Y. Ding, **X.-C. Wu**, A. Holmes, A. Wiseth, D. Franklin, M. Martonosi, and F. T. Chong, "SQUARE: Strategic Quantum Ancilla Reuse for Modular Quantum Programs via Cost-Effective Uncomputation", in *Proceedings of 47th Intl. Symposium on Computer Architecture (ISCA)*, **Award: Honorable Mention for IEEE Micro Top Picks**, May 2020.

[3] **X.-C. Wu**, M. G. Davis, F. T. Chong, and C. Iancu, *Optimizing noisy-intermediate scale quantum circuits: A block-based synthesis*, 2020. arXiv: 2012.09835 [quant-ph].

[4] F. Cappello, S. Di, S. Li, X. Liang, A. M. Gok, D. Tao, C. H. Yoon, **X.-C. Wu**, Y. Alexeev, and F. T. Chong, "Use cases of lossy compression for floating-point data in scientific data sets", *The International Journal of High Performance Computing Applications*, vol. 33, no. 6, pp. 1201–1220, 2019.

[5] **X.-C. Wu**, S. Di, F. Cappello, H. Finkel, Y. Alexeev, and F. Chong, "Intermediate-scale full state quantum circuit simulation by using lossy data compression", *Bulletin of the American Physical Society*, vol. 64, Mar. 2019.

[6] **X.-C. Wu**, S. Di, E. M. Dasgupta, F. Cappello, H. Finkel, Y. Alexeev, and F. T. Chong, "Full-state quantum circuit cimulation by using data compression", in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Nov. 2019, pp. 1–24.

[7] **X.-C. Wu**, Y. Ding, Y. Shi, Y. Alexeev, H. Finkel, K. Kim, and F. T. Chong, "ILP-Based Scheduling for Linear-Tape Model Trapped-Ion Quantum Computers", in *IEEE/ACM 30th The International Conference for High Performance Computing, Networking, Storage and Analysis (SC) [Poster]*, Nov. 2019.

[8] **X.-C. Wu**, T. Sherwood, F. T. Chong, and Y. Li, "Protecting page tables from rowhammer attacks using monotonic pointers in dram true-cells", in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2019, pp. 645–657.

[9] A. Javadi-Abhari, A. Holmes, S. Patil, J. Heckey, D. Kudrow, P. Gokhale, D. Noursi, L. Ehudin, Y. Ding, **X.-C. Wu**, and Y. Shi, "Scaffcc: Scaffold compiler collection", 2018.

[10] **X.-C. Wu**, S. Di, F. Cappello, H. Finkel, Y. Alexeev, and F. T. Chong, "Amplitude-aware lossy compression for quantum circuit simulation", in *The 4th International Workshop on Data Reduction for Big Scientific Data (DRBSD-4) in conjunction with IEEE/ACM 29th The International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2018.

[11] **X.-C. Wu**, S. Di, F. Cappello, H. Finkel, Y. Alexeev, and F. T. Chong, "Full State Quantum Circuit Simulation by Using Data Compression", in *IEEE/ACM 29th The International Conference for High Performance Computing, Networking, Storage and Analysis (SC) [Poster]*, Nov. 2018.

[12] **X.-C. Wu**, S. Di, F. Cappello, H. Finkel, Y. Alexeev, and F. T. Chong, "Memory-efficient quantum circuit simulation by using lossy data compression", in *The 3rd International Workshop on Post-Moore Era Supercomputing (PMES) in conjunction with IEEE/ACM 29th The International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2018.

Teaching

- **Teaching Assistant** Tutorial: Grand Challenges and Research Tools for Quantum Computing
  Jun 2019. DAC in Las Vegas, NV
  Mar 2019. DATE in Florence, Italy
  Oct 2018. MICRO in Fukuoka, Japan
  Jun 2018. ISCA in Los Angeles, CA
- **Teaching Assistant** at University of Chicago — Autumn 2017
  Computer Architecture
- **Teaching Assistant** at University of Chicago — Winter 2017
  Mobile Computing
- **Teaching Assistant** at University of Chicago — Autumn 2016
  Computer Architecture

Invited Talks and Presentations

- **Scalable Quantum Circuit Optimization Using Automated Synthesis** Mar 2021
  Conference talk at APS March Meeting
- **TILT:Achieving Higher Fidelity on a Trapped-Ion Linear-Tape Architecture** Feb 2021
  Conference talk at HPCA
- **Design, Optimization, and Simulation of Scalable Quantum Computing Systems** Jan 2021
  Invited talk at IonQ
- **Quantum Circuit Optimization by Using Scalable Quantum Synthesis** Sep 2020
  Invited seminar talk at AQT, Lawrence Berkeley National Laboratory, Berkeley, CA (Virtual)
- **Full-State Quantum Circuit Simulation by Using Data Compression** Nov 2019
  Conference talk at SC19, Denver, CO
- **ILP-Based Scheduling for Linear-Tape Model Trapped-Ion Quantum Computers** Nov 2019
  Poster presentation at SC19, Denver, CO
- **ILP-Based Scheduling for Linear-Tape Model Trapped-Ion Quantum Computers** Sep 2019
  Student seminar talk at Argonne National Laboratory, Lemont, IL
- **ILP-Based Scheduling for Linear-Tape Model Trapped-Ion Quantum Computers** Aug 2019
  Talk at QIS student workshop, Argonne National Laboratory, Lemont, IL
- **Full-State Quantum Circuit Simulation by Using Data Compression** Apr 2019
  Master presentation at University of Chicago, Chicago, IL
- **Protecting Page Tables from RowHammer Attacks using DRAM True-Cells** Apr 2019
  Conference Talk at ASPLOS, Providence, RI
- **Full State Quantum Circuit Simulation by Using Lossy Data Compression** Mar 2019
  Talk at APS March Meeting, Boston, MA
- **Memory-Efficient Quantum Circuit Simulation by Using Lossy Data Compression** Nov 2018
  Talk at Workshop on Post-Moore Era Supercomputing (PMES), Dallas, TX
- **Amplitude-Aware Lossy Compression for Quantum Circuit Simulation** Nov 2018
  Talk at Workshop on Data Reduction for Big Scientific Data (DRBSD-4), Dallas, TX

Scholarships and Awards

- IEEE Micro Top Picks in Computer Architecture of 2020 2021
  for the paper SQUARE: Strategic Quantum Ancilla Reuse for Modular Quantum Programs via Cost-Effective Uncomputation with Y. Ding et al.
- ASPLOS Student travel grant Providence, RI 2019
- MICRO Student travel grant Fukuoka, Japan 2018
- ISCA Student travel grant Los Angeles, CA 2018
- CAV Student travel grant Heidelberg, German 2017
- ISCA Student travel grant Toronto, Canada 2017

REFERENCES

[1] Scott Aaronson and Lijie Chen. Complexity-theoretic foundations of quantum supremacy experiments. *arXiv preprint arXiv:1612.05903*, 2016.

[2] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5):052328, 2004.

[3] Ola Al-Ta'ani. *Quantum circuit synthesis using Solovay-Kitaev algorithm and optimization techniques*. PhD thesis, Kansas State University, 2015.

[4] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, Luciano Bello, Yael Ben-Haim, D Bucher, FJ Cabrera-Hernández, J Carballo-Franquis, A Chen, CF Chen, et al. Qiskit: An open-source framework for quantum computing. *Accessed on: Mar*, 16, 2019.

[5] Mohammad AlFailakawi, Imtiaz Ahmad, and Suha Hamdan. Lnn reversible circuit realization using fast harmony search based heuristic. In *Asia-Pacific Conference on Computer Science and Electrical Engineering*, 2014.

[6] Zaid Alwardi, Robert Wille, and Rolf Drechsler. Synthesis of reversible circuits using conventional hardware description languages. In *2018 IEEE 48th International Symposium on Multiple-Valued Logic (ISMVL)*, pages 97–102. IEEE, 2018.

[7] Matthew Amy, Dmitri Maslov, Michele Mosca, and Martin Roetteler. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(6):818–830, 2013.

[8] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[9] Ryo Asai. Mcdram as high-bandwidth memory (hbm) in knights landing processors: developers guide. *Colfax International*, 2016.

[10] Costin Bădescu, Ryan O'Donnell, and John Wright. Quantum state certification. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 503–514, 2019.

[11] Tayebeh Bahreini and Naser Mohammadzadeh. An MINLP model for scheduling and placement of quantum circuits with a heuristic solution approach. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 12(3):1–20, 2015.

[12] Harrison Ball, Michael J Biercuk, Andre Carvalho, Rajib Chakravorty, Jiayin Chen, Leonardo A de Castro, Steven Gore, David Hover, Michael Hush, Per J Liebermann, et al. Software tools for quantum control: Improving quantum computer performance through noise and error suppression. *arXiv preprint arXiv:2001.04060*, 2020.

[13] C. J. Ballance, T. P. Harty, N. M. Linke, M. A. Sepiol, and D. M. Lucas. High-fidelity quantum logic gates using trapped-ion hyperfine qubits. *Phys. Rev. Lett.*, 117:060504, Aug 2016.

[14] Chandan Bandyopadhyay, Robert Wille, Rolf Drechsler, and Hafizur Rahaman. Post synthesis-optimization of reversible circuit using template matching. In *2020 24th International Symposium on VLSI Design and Test (VDAT)*, pages 1–4. IEEE, 2020.

[15] Rami Barends, Julian Kelly, Anthony Megrant, Andrzej Veitia, Daniel Sank, Evan Jeffrey, Ted C White, Josh Mutus, Austin G Fowler, Brooks Campbell, et al. Superconducting quantum circuits at the surface code threshold for fault tolerance. *Nature*, 508(7497):500, 2014.

[16] Rami Barends, Alireza Shabani, Lucas Lamata, Julian Kelly, Antonio Mezzacapo, Urtzi Las Heras, Ryan Babbush, Austin G Fowler, Brooks Campbell, Yu Chen, et al. Digitized adiabatic quantum computing with a superconducting circuit. *Nature*, 534(7606):222–226, 2016.

[17] MD Barrett, J Chiaverini, T Schaetz, J Britton, WM Itano, JD Jost, E Knill, C Langer, D Leibfried, R Ozeri, et al. Deterministic quantum teleportation of atomic qubits. *Nature*, 429(6993):737, 2004.

[18] Charles H Bennett, David P DiVincenzo, John A Smolin, and William K Wootters. Mixed-state entanglement and quantum error correction. *Physical Review A*, 54(5):3824, 1996.

[19] Charles H. Bennett, David P. DiVincenzo, John A. Smolin, and William K. Wootters. Mixed-state entanglement and quantum error correction. *Phys. Rev. A*, 54:3824–3851, Nov 1996.

[20] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on computing*, 26(5):1411–1473, 1997.

[21] Anirban Bhattacharjee, Chandan Bandyopadhyay, Robert Wille, Rolf Drechsler, and Hafizur Rahaman. A novel approach for nearest neighbor realization of 2D quantum circuits. In *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 305–310. IEEE, 2018.

[22] Jacob Biamonte and Ville Bergholm. Tensor networks in a nutshell. *arXiv preprint arXiv:1708.00006*, 2017.

[23] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195, 2017.

[24] RB Blakestad, C Ospelkaus, AP VanDevender, JM Amini, J Britton, D Leibfried, and David J Wineland. High-fidelity transport of trapped-ion qubits through an X-junction trap array. *Physical Review Letters*, 102(15):153002, 2009.

[25] BlosC compressor. http://blosc.org/. Online.

[26] Reinhold Blumel, Nikodem Grzesiak, and Yunseong Nam. Power-optimal, stabilized entangling gate between trapped-ion qubits. *arXiv preprint arXiv:1905.09292*, 2019.

[27] Alex Bocharov, Martin Roetteler, and Krysta M Svore. Efficient synthesis of universal repeat-until-success quantum circuits. *Physical review letters*, 114(8):080502, 2015.

[28] Sergio Boixo, Sergei V Isakov, Vadim N Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, Michael J Bremner, John M Martinis, and Hartmut Neven. Characterizing quantum supremacy in near-term devices. *Nature Physics*, 14(6):595, 2018.

[29] Sergio Boixo, Sergei V Isakov, Vadim N Smelyanskiy, and Hartmut Neven. Simulation of low-depth quantum circuits as complex undirected graphical models. *arXiv preprint arXiv:1712.05384*, 2017.

[30] X Bonet-Monroig, R Sagastizabal, M Singh, and TE O'Brien. Low-cost error mitigation by symmetry verification. *Physical Review A*, 98(6):062339, 2018.

[31] Gilles Brassard. Teleportation as a quantum computation. *arXiv preprint quant-ph/9605035*, 1996.

[32] Sergey Bravyi and David Gosset. Improved classical simulation of quantum circuits dominated by clifford gates. *Physical review letters*, 116(25):250501, 2016.

[33] Sergey Bravyi, David Gosset, and Robert König. Quantum advantage with shallow circuits. *Science*, 362(6412):308–311, 2018.

[34] Ken R Brown. Software-tailored architectures for quantum codesign. 2019.

[35] Kenneth R Brown, Aram W Harrow, and Isaac L Chuang. Arbitrarily accurate composite pulse sequences. *Physical Review A*, 70(5):052318, 2004.

[36] Natalie C. Brown and Kenneth R. Brown. Comparing Zeeman qubits to hyperfine qubits in the context of the surface code: $^{174}Yb^{+}$ and $^{171}Yb^{+}$. *Phys. Rev. A*, 97:052301, May 2018.

[37] Colin D Bruzewicz, John Chiaverini, Robert McConnell, and Jeremy M Sage. Trapped-ion quantum computing: Progress and challenges. *Applied Physics Reviews*, 6(2):021314, 2019.

[38] M. Burtscher and P. Ratanaworabhan. FPC: A high-speed compressor for double-precision floating-point data. *IEEE Transactions on Computers*, 58(1):18–31, Jan 2009.

[39] Daan Camps and Roel Van Beeumen. Approximate quantum circuit synthesis using block encodings. *Physical Review A*, 102(5):052411, 2020.

[40] Amlan Chakrabarti, Susmita Sur-Kolay, and Ayan Chaudhury. Linear nearest neighbor synthesis of reversible circuits by graph partitioning. *arXiv preprint arXiv:1112.0564*, 2011.

[41] Jianxin Chen, Fang Zhang, Mingcheng Chen, Cupjin Huang, Michael Newman, and Yaoyun Shi. Classical simulation of intermediate-size quantum circuits. *arXiv preprint arXiv:1805.01450*, 2018.

[42] Zhao-Yun Chen, Qi Zhou, Cheng Xue, Xia Yang, Guang-Can Guo, and Guo-Ping Guo. 64-qubit quantum circuit simulation. *Science Bulletin*, 2018.

[43] Jinglei Cheng, Haoqing Deng, and Xuehai Qian. Accqoc: Accelerating quantum optimal control based pulse generation. In *47th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2020, Valencia, Spain, May 30 - June 3, 2020*, pages 543–555. IEEE, 2020.

[44] Richard Cleve, Artur Ekert, Chiara Macchiavello, and Michele Mosca. Quantum algorithms revisited. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1969):339–354, 1998.

[45] John Clyne, Pablo Mininni, Alan Norton, and Mark Rast. Interactive desktop analysis of high resolution simulations: application to turbulent plume dynamics and current sheet formation. *New Journal of Physics*, 9(301):1–29, 2007.

[46] Yann Collet. Zstandard - real-time data compression algorithm. *http://facebook.github.io/zstd/*, 2015.

[47] Alexander Cowtan, Silas Dilkes, Ross Duncan, Will Simmons, and Seyon Sivarajah. Phase gadget synthesis for shallow circuits. *arXiv preprint arXiv:1906.01734*, 2019.

[48] Andrew Cross. The IBM Q experience and QISKit open-source quantum computing software. *Bulletin of the American Physical Society*, 63, 2018.

[49] Steven A Cuccaro, Thomas G Draper, Samuel A Kutin, and David Petrie Moulton. A new quantum ripple-carry addition circuit. *arXiv preprint quant-ph/0410184*, 2004.

[50] Pierre-Luc Dallaire-Demers and Nathan Killoran. Quantum generative adversarial networks. *Physical Review A*, 98(1):012324, 2018.

[51] Marc Grau Davis, Ethan Smith, Ana Tudor, Koushik Sen, Irfan Siddiqi, and Costin Iancu. Heuristics for quantum compiling with a continuous gate set. *arXiv preprint arXiv:1912.02727*, 2019.

[52] Christopher M Dawson and Michael A Nielsen. The solovay-kitaev algorithm. *arXiv preprint quant-ph/0505030*, 2005.

[53] Koen De Raedt, Kristel Michielsen, Hans De Raedt, Binh Trieu, Guido Arnold, Marcus Richter, Th Lippert, H Watanabe, and N Ito. Massively parallel quantum computer simulator. *Computer Physics Communications*, 176(2):121–136, 2007.

[54] Alexis De Vos and Stijn De Baerdemacker. Block-z x z synthesis of an arbitrary quantum circuit. *Physical Review A*, 94(5):052317, 2016.

[55] Shantanu Debnath, Norbert M Linke, Caroline Figgatt, Kevin A Landsman, Kevin Wright, and Christopher Monroe. Demonstration of a small programmable quantum computer with atomic qubits. *Nature*, 536(7614):63–66, 2016.

[56] L Peter Deutsch. GZIP file format specification version 4.3, 1996.

[57] Sheng Di and Franck Cappello. Fast error-bounded lossy HPC data compression with SZ. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 730–739. IEEE, 2016.

[58] Sheng Di and Franck Cappello. Fast error-bounded lossy HPC data compression with SZ. In *IPDPS 2016*, pages 730–739, 2016.

[59] Olivia Di Matteo and Michele Mosca. Parallelizing quantum circuit synthesis. *Quantum Science and Technology*, 1(1):015003, 2016.

[60] Yongshan Ding, Xin-Chuan Wu, Adam Holmes, Ash Wiseth, Diana Franklin, Margaret Martonosi, and Frederic T Chong. Square: Strategic quantum ancilla reuse for modular quantum programs via cost-effective uncomputation. In *ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, 2020.

[61] David P DiVincenzo. Two-bit gates are universal for quantum computation. *Physical Review A*, 51(2):1015, 1995.

[62] Suguru Endo, Simon C Benjamin, and Ying Li. Practical quantum error mitigation for near-future applications. *Physical Review X*, 8(3):031027, 2018.

[63] Argonne Leadership Computing Facility. Aurora supercomputer. https://www.alcf.anl.gov/programs/aurora-esp, 2019. Online. Accessed: 2019-04-06.

[64] SD Fallek, CD Herold, BJ McMahon, KM Maller, KR Brown, and JM Amini. Transport implementation of the Bernstein–Vazirani algorithm with ion qubits. *New Journal of Physics*, 18(8):083030, 2016.

[65] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[66] Austin G Fowler, Matteo Mariantoni, John M Martinis, and Andrew N Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3):032324, 2012.

[67] C Gidney and D Bacon. The cirq developers, quantumlib/-cirq: A python framework for creating, editing, and invoking noisy intermediate scale quantum (nisq) circuits.

[68] Brett Giles and Peter Selinger. Exact synthesis of multiqubit clifford+ t circuits. *Physical Review A*, 87(3):032332, 2013.

[69] Steffen J Glaser, Ugo Boscain, Tommaso Calarco, Christiane P Koch, Walter Köckenberger, Ronnie Kosloff, Ilya Kuprov, Burkhard Luy, Sophie Schirmer, Thomas Schulte-Herbrüggen, et al. Training schrödinger's cat: quantum optimal control. *The European Physical Journal D*, 69(12):1–24, 2015.

[70] Pranav Gokhale, Yongshan Ding, Thomas Propson, Christopher Winkler, Nelson Leung, Yunong Shi, David I Schuster, Henry Hoffmann, and Frederic T Chong. Partial compilation of variational algorithms for noisy intermediate-scale quantum machines. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 266–278, 2019.

[71] Pranav Gokhale, Ali Javadi-Abhari, Nathan Earnest, Yunong Shi, and Frederic T Chong. Optimized quantum compilation for near-term algorithms with openpulse. *arXiv preprint arXiv:2004.11205*, 2020.

[72] Leslie Ann Goldberg and Heng Guo. The complexity of approximating complex-valued Ising and Tutte partition functions. *computational complexity*, 26(4):765–833, 2017.

[73] A Preview of Bristlecone, Google's New Quantum Processor. https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html. Accessed: 2020-10-09.

[74] Daniel Gottesman. Stabilizer codes and quantum error correction. *arXiv preprint quant-ph/9705052*, 1997.

[75] Daniel Gottesman. The Heisenberg representation of quantum computers. *arXiv preprint quant-ph/9807006*, 1998.

[76] Daniel Gottesman. An introduction to quantum error correction and fault-tolerant quantum computation. In *Quantum information science and its contributions to mathematics, Proceedings of Symposia in Applied Mathematics*, volume 68, pages 13–58, 2010.

[77] William D Gropp, William Gropp, Ewing Lusk, and Anthony Skjellum. *Using MPI: portable parallel programming with the message-passing interface*, volume 1. MIT press, 1999.

[78] Daniel Große, Robert Wille, Gerhard W Dueck, and Rolf Drechsler. Exact multiple-control toffoli network synthesis with sat techniques. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(5):703–715, 2009.

[79] Lov K Grover. A fast quantum mechanical algorithm for database search. *arXiv preprint quant-ph/9605043*, 1996.

[80] Nikodem Grzesiak, Reinhold Blümel, Kristin Beck, Kenneth Wright, Vandiver Chaplin, Jason M Amini, Neal C Pisenti, Shantanu Debnath, Jwo-Sy Chen, and Yunseong Nam. Efficient arbitrary simultaneously entangling gates on a trapped-ion quantum computer. *arXiv preprint arXiv:1905.09294*, 2019.

[81] M. Gutiérrez, M. Müller, and A. Bermúdez. Transversality and lattice surgery: Exploring realistic routes toward coupled logical qubits with trapped-ion quantum processors. *Phys. Rev. A*, 99:022330, Feb 2019.

[82] Thomas Häner and Damian S Steiger. 0.5 petabyte simulation of a 45-qubit quantum circuit. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, page 33. ACM, 2017.

[83] Fred C Hennie. One-tape, off-line Turing machine computations. *Information and Control*, 8(6):553–578, 1965.

[84] WK Hensinger, S Olmschenk, D Stick, D Hucul, M Yeo, M Acton, L Deslauriers, C Monroe, and J Rabchuk. T-junction ion trap array for two-dimensional ion shuttling, storage, and manipulation. *Applied Physics Letters*, 88(3):034101, 2006.

[85] Markus Heyl, Anatoli Polkovnikov, and Stefan Kehrein. Dynamical quantum phase transitions in the transverse-field ising model. *Physical review letters*, 110(13):135704, 2013.

[86] Ling Hu, Shu-Hao Wu, Weizhou Cai, Yuwei Ma, Xianghao Mu, Yuan Xu, Haiyan Wang, Yipu Song, Dong-Ling Deng, Chang-Ling Zou, et al. Quantum generative adversarial learning in a superconducting quantum circuit. *Science advances*, 5(1):eaav2761, 2019.

[87] Yipeng Huang and Margaret Martonosi. Statistical assertions for validating patterns and finding bugs in quantum programs". In *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2019.

[88] G Huber, T Deuschle, W Schnitzler, R Reichle, K Singer, and F Schmidt-Kaler. Transport of ions in a segmented linear paul trap in printed-circuit-board technology. *New Journal of Physics*, 10(1):013004, 2008.

[89] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, Sep. 1952.

[90] Lawrence Ibarria, Peter Lindstrom, Jarek Rossignac, and Andrzej Szymczak. Out-of-core compression and decompression of large n-dimensional scalar fields. *Computer Graphics Forum*, 22(3):343–348, 2003.

[91] IBM Announces Advances to IBM Quantum Systems and Ecosystem. https://www-03.ibm.com/press/us/en/pressrelease/53374.wss. Accessed: 2020-10-09.

[92] IBM Quantum Experience. https://quantum-computing.ibm.com/. Accessed: 2020-11-15.

[93] IONQ. IonQ harnesses single-atom qubits to build the world's most powerful quantum computer. https://ionq.com/news/december-11-2018, 2018. Online.

[94] Toshinari Itoko, Rudy Raymond, Takashi Imamichi, and Atsushi Matsuo. Optimization of quantum circuit mapping using gate transformation and commutation. *Integration*, 70:43–50, 2020.

[95] Ali JavadiAbhari, Shruti Patil, Daniel Kudrow, Jeff Heckey, Alexey Lvov, Frederic T Chong, and Margaret Martonosi. ScaffCC: Scalable compilation and analysis of quantum programs. *Parallel Computing*, 45:2–17, 2015.

[96] James Jeffers and James Reinders. *Intel Xeon Phi coprocessor high performance programming*. Newnes, 2013.

[97] Richard Jozsa. Quantum factoring, discrete logarithms, and the hidden subgroup problem. *Computing in science & engineering*, 3(2):34, 2001.

[98] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242, 2017.

[99] Abhinav Kandala, Kristan Temme, Antonio D Córcoles, Antonio Mezzacapo, Jerry M Chow, and Jay M Gambetta. Error mitigation extends the computational reach of a noisy quantum processor. *Nature*, 567(7749):491, 2019.

[100] H Kaufmann, T Ruster, Christian Tomás Schmiegelow, Marcelo Alejandro Luda, V Kaushal, J Schulz, D Von Lindenfels, F Schmidt-Kaler, and UG Poschinger. Scalable creation of long-lived multipartite entanglement. *Physical Review Letters*, 119(15):150503, 2017.

[101] Julian Kelly. A preview of Bristlecone, Google's new quantum processor. *Google Research Blog*, 5, 2018.

[102] D. Kielpinski, C. Monroe, and D. J. Wineland. Architecture for a large-scale ion-trap quantum computer. *Nature*, 417(6890):709–711, 2002.

[103] Jungsang Kim, Peter Maunz, Taehyun Kim, Jeffrey Hussman, Rachel Noek, Abhijit Mehta, and Christopher Monroe. Modular universal scalable ion-trap quantum computer (MUSIQC). 1363(1):190–193, 2011.

[104] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Fast and efficient exact synthesis of single qubit unitaries generated by clifford and t gates. *arXiv preprint arXiv:1206.5236*, 2012.

[105] Abhoy Kole, Kamalika Datta, and Indranil Sengupta. A heuristic for linear nearest neighbor realization of quantum circuits by SWAP gate insertion using N-gate lookahead. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 6(1):62–72, 2016.

[106] Sriram Lakshminarasimhan, Neil Shah, Stephane Ethier, Seung-Hoe Ku, Choong-Seock Chang, Scott Klasky, Rob Latham, Rob Ross, and Nagiza F Samatova. Isabela for effective in situ compression of scientific data. *Concurrency and Computation: Practice and Experience*, 25(4):524–540, 2013.

[107] Bjoern Lekitsch, Sebastian Weidt, Austin G Fowler, Klaus Mølmer, Simon J Devitt, Christof Wunderlich, and Winfried K Hensinger. Blueprint for a microwave trapped ion quantum computer. *Science Advances*, 3(2):e1601540, 2017.

[108] Pak Hong Leung and Kenneth R Brown. Entangling an arbitrary pair of qubits in a long ion crystal. *Physical Review A*, 98(3):032318, 2018.

[109] Pak Hong Leung, Kevin A Landsman, Caroline Figgatt, Norbert M Linke, Christopher Monroe, and Kenneth R Brown. Robust 2-qubit gates in a linear ion crystal using a frequency-modulated driving force. *Physical review letters*, 120(2):020501, 2018.

[110] Gushu Li, Yufei Ding, and Yuan Xie. Tackling the qubit mapping problem for nisq-era quantum devices. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1001–1014, 2019.

[111] Riling Li, Bujiao Wu, Mingsheng Ying, Xiaoming Sun, and Guangwen Yang. Quantum supremacy circuit simulation on Sunway taihuLight. *arXiv preprint arXiv:1804.04797*, 2018.

[112] X. Liang, S. Di, D. Tao, S. Li, S. Li, H. Guo, Z. Chen, and F. Cappello. Error-controlled lossy compression optimized for high compression ratios of scientific datasets. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 438–447, Dec 2018.

[113] Xin Liang, Sheng Di, Dingwen Tao, Zizhong Chen, and Franck Cappello. An efficient transformation scheme for lossy data compression with point-wise relative error bound. In *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 179–189. IEEE, 2018.

[114] G-D Lin, S-L Zhu, Rajibul Islam, Kihwan Kim, M-S Chang, Simcha Korenblit, Christopher Monroe, and L-M Duan. Large-scale quantum computation in an anharmonic linear ion trap. *EPL (Europhysics Letters)*, 86(6):60004, 2009.

[115] Peter Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2674–2683, 2014.

[116] Peter Lindstrom and Martin Isenburg. Fast and efficient compression of floating-point data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1245–1250, 2006.

[117] Norbert M Linke, Dmitri Maslov, Martin Roetteler, Shantanu Debnath, Caroline Figgatt, Kevin A Landsman, Kenneth Wright, and Christopher Monroe. Experimental comparison of two quantum computing architectures. *Proceedings of the National Academy of Sciences*, 114(13):3305–3310, 2017.

[118] Seth Lloyd and Christian Weedbrook. Quantum generative adversarial learning. *Physical Review Letters*, 121(4):040502, 2018.

[119] T. Lu, Q. Liu, X. He, H. Luo, E. Suchyta, J. Choi, N. Podhorszki, S. Klasky, M. Wolf, T. Liu, and Z. Qiao. Understanding and modeling lossy compression schemes on HPC scientific data. In *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 348–357, May 2018.

[120] Austin P Lund, Michael J Bremner, and Timothy C Ralph. Quantum sampling problems, bosonsampling and quantum supremacy. *npj Quantum Information*, 3(1):1–8, 2017.

[121] Aaron Lye, Robert Wille, and Rolf Drechsler. Determining the minimal number of swap gates for multi-dimensional nearest neighbor quantum circuits. In *The 20th Asia and South Pacific Design Automation Conference*, pages 178–183. IEEE, 2015.

[122] Igor L Markov, Aneeqa Fatima, Sergei V Isakov, and Sergio Boixo. Quantum supremacy is both closer and farther than it appears. *arXiv preprint arXiv:1807.10749*, 2018.

[123] Igor L Markov and Yaoyun Shi. Simulating quantum computation by contracting tensor networks. *SIAM Journal on Computing*, 38(3):963–981, 2008.

[124] Esteban A Martinez, Thomas Monz, Daniel Nigg, Philipp Schindler, and Rainer Blatt. Compiling quantum algorithms for architectures with multi-qubit gates. *New Journal of Physics*, 18(6):063029, 2016.

[125] Dmitri Maslov. Basic circuit compilation techniques for an ion-trap quantum machine. *New Journal of Physics*, 19(2):023035, 2017.

[126] Dmitri Maslov, Sean M Falconer, and Michele Mosca. Quantum circuit placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(4):752–763, 2008.

[127] Alexander McCaskey, Eugene Dumitrescu, Mengsu Chen, Dmitry Lyakh, and Travis Humble. Validating quantum-classical programming models with tensor network simulations. *PloS one*, 13(12):e0206704, 2018.

[128] Alistair R Milne, Claire L Edmunds, Cornelius Hempel, Federico Roy, Sandeep Mavadia, and Michael J Biercuk. Phase-modulated entangling gates robust to static and time-varying errors. *Physical Review Applied*, 13(2):024022, 2020.

[129] Nikolaj Moll, Panagiotis Barkoutsos, Lev S Bishop, Jerry M Chow, Andrew Cross, Daniel J Egger, Stefan Filipp, Andreas Fuhrer, Jay M Gambetta, Marc Ganzhorn, et al. Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Science and Technology*, 3(3):030503, 2018.

[130] C Monroe, R Raussendorf, A Ruthven, KR Brown, P Maunz, L-M Duan, and J Kim. Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects. *Physical Review A*, 89(2):022317, 2014.

[131] Christopher Monroe. 2019 CLEO Plenary: Christopher Monroe: Quantum Computing with Atoms. 2019.

[132] M Mudrich, S Kraft, K Singer, R Grimm, A Mosk, and M Weidemüller. Sympathetic cooling with two atomic species in an optical trap. *Physical review letters*, 88(25):253001, 2002.

[133] Prakash Murali, Dripto M Debroy, Kenneth R Brown, and Margaret Martonosi. Architecting noisy intermediate-scale trapped ion quantum computers. *arXiv preprint arXiv:2004.04706*, 2020.

[134] Prakash Murali, Norbert Matthias Linke, Margaret Martonosi, Ali Javadi Abhari, Nhung Hong Nguyen, and Cinthia Huerta Alderete. Full-stack, real-system quantum computer studies: architectural comparisons and design insights. In *Proceedings of the 46th International Symposium on Computer Architecture*, pages 527–540, 2019.

[135] Attila B Nagy. On an implementation of the solovay-kitaev algorithm. *arXiv preprint quant-ph/0606077*, 2006.

[136] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[137] Philipp Niemann, Robert Wille, and Rolf Drechsler. Improved synthesis of clifford+ t quantum functionality. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 597–600. IEEE, 2018.

[138] Philipp Niemann, Robert Wille, and Rolf Drechsler. Advanced exact synthesis of clifford+ t circuits. *Quantum Information Processing*, 19(9):1–23, 2020.

[139] Shin Nishio, Yulu Pan, Takahiko Satoh, Hideharu Amano, and Rodney Van Meter. Extracting success from ibm's 20-qubit machines using error-aware compilation. *arXiv preprint arXiv:1903.10963*, 2019.

[140] Roee Ozeri. The trapped-ion qubit tool box. *Contemporary Physics*, 52(6):531–550, 2011.

[141] Scott Parker, Vitali Morozov, Sudheer Chunduri, Kevin Harms, Chris Knight, and Kalyan Kumaran. Early evaluation of the Cray XC40 Xeon Phi system 'Theta' at Argonne. Technical report, Argonne National Lab.(ANL), Argonne, IL (United States), 2017.

[142] Edwin Pednault, John A Gunnels, Giacomo Nannicini, Lior Horesh, Thomas Magerlein, Edgar Solomonik, and Robert Wisnieff. Breaking the 49-qubit barrier in the simulation of quantum circuits. *arXiv preprint arXiv:1710.05867*, 2017.

[143] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5:4213, 2014.

[144] JM Pino, JM Dreiling, C Figgatt, JP Gaebler, SA Moses, CH Baldwin, M Foss-Feig, D Hayes, K Mayer, C Ryan-Anderson, et al. Demonstration of the QCCD trapped-ion quantum computer architecture. *arXiv preprint arXiv:2003.01293*, 2020.

[145] John Preskill. Quantum computing in the NISQ era and beyond. *Quantum*, 2:79, 2018.

[146] Quantiki. A list of qc simulators. https://quantiki.org/wiki/list-qc-simulators, 2019. Online.

[147] Rigetti Announces Its Hybrid Quantum Computing Platform and a 1m Prize. https://techcrunch.com/2018/09/07/rigetti-announces-its-hybrid-quantum-computing-platform-and-a-1m-prize/. Accessed: 2019-03-29.

[148] Mary A Rowe, Amit Ben-Kish, Brian Demarco, Dietrich Leibfried, Volker Meyer, Jim Beall, Joe Britton, J Hughes, Wayne M Itano, Brana Jelenkovic, et al. Transport of quantum states and separation of ions in a dual RF ion trap. *Quantum Information and Computation*, 2(4):15, 2002.

[149] Mehdi Saeedi, Robert Wille, and Rolf Drechsler. Synthesis of quantum circuits for linear nearest neighbor architectures. *Quantum Information Processing*, 10(3):355–377, 2011.

[150] Ramiro Sagastizabal, Xavier Bonet-Monroig, Malay Singh, MA Rol, CC Bultink, X Fu, CH Price, VP Ostroukh, N Muthusubramanian, A Bruno, et al. Error mitigation by symmetry verification on a variational quantum eigensolver. *arXiv preprint arXiv:1902.11258*, 2019.

[151] Sahar Sargaran and Naser Mohammadzadeh. SAQIP: A scalable architecture for quantum information processors. *ACM Transactions on Architecture and Code Optimization (TACO)*, 16(2):1–21, 2019.

[152] Naoto Sasaki, Kento Sato, Toshio Endo, and Satoshi Matsuoka. Exploration of lossy compression for application-level checkpoint/restart. In *IPDPS 2015*, pages 914–922, 2015.

[153] Vivek V Shende, Stephen S Bullock, and Igor L Markov. Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1000–1010, 2006.

[154] Yunong Shi, Nelson Leung, Pranav Gokhale, Zane Rossi, David I Schuster, Henry Hoffmann, and Frederic T Chong. Optimized compilation of aggregated instructions for realistic quantum computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1031–1044, 2019.

[155] Dongbin Shin, Hannes Hübener, Umberto De Giovannini, Hosub Jin, Angel Rubio, and Noejung Park. Phonon-driven spin-floquet magneto-valleytronics in mos 2. *Nature communications*, 9(1):1–8, 2018.

[156] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[157] Marcos Yukio Siraichi, Vinícius Fernandes dos Santos, Sylvain Collange, and Fernando Magno Quintão Pereira. Qubit allocation. In *Proceedings of the 2018 International Symposium on Code Generation and Optimization*, pages 113–125, 2018.

[158] Marcos Yukio Siraichi, Vin'icius Fernandes dos Santos, Sylvain Collange, and Fernando Magno Quintao Pereira. Qubit allocation. In *Proceedings of the 2018 International Symposium on Code Generation and Optimization*, pages 113–125, 2018.

[159] Haozhen Situ, Zhimin He, Lvzhou Li, and Shenggen Zheng. Quantum generative adversarial network for generating discrete data. *arXiv preprint arXiv:1807.01235*, 2018.

[160] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington, and Ross Duncan. t— ket¿: A retargetable compiler for nisq devices. *Quantum Science and Technology*, 2020.

[161] Mikhail Smelyanskiy, Nicolas PD Sawaya, and Alán Aspuru-Guzik. qhipster: the quantum high performance software testing environment. *arXiv preprint arXiv:1601.07195*, 2016.

[162] Avinash Sodani. Knights Landing (KNL): 2nd generation intel® xeon phi processor. In *2015 IEEE Hot Chips 27 Symposium (HCS)*, pages 1–24. IEEE, 2015.

[163] Dan Stick, WK Hensinger, Steven Olmschenk, MJ Madsen, Keith Schwab, and Chris Monroe. Ion trap in a semiconductor chip. *Nature Physics*, 2(1):36, 2006.

[164] Erich Strohmaier, Jack Dongarra, Horst Simon, Martin Meuer, and Hans Meuer. TOP500 list. 2018.

[165] Dingwen Tao, Sheng Di, Zizhong Chen, and Franck Cappello. Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization. In *Parallel and Distributed Processing Symposium (IPDPS), 2017 IEEE International*, pages 1129–1139. IEEE, 2017.

[166] N Timoney, I Baumgart, M Johanning, AF Varón, Martin B Plenio, A Retzker, and Ch Wunderlich. Quantum gates and memory using microwave-dressed states. *Nature*, 476(7359):185–188, 2011.

[167] Colin J Trout, Muyuan Li, Mauricio Gutiérrez, Yukai Wu, Sheng-Tao Wang, Luming Duan, and Kenneth R Brown. Simulating the performance of a distance-3 surface code in a linear ion trap. *New Journal of Physics*, 20(4):043038, 2018.

[168] Robert R Tucci. An introduction to cartan's kak decomposition for qc programmers. *arXiv preprint quant-ph/0507171*, 2005.

[169] Benjamin Villalonga, Dmitry Lyakh, Sergio Boixo, Hartmut Neven, Travis S Humble, Rupak Biswas, Eleanor Rieffel, Alan Ho, and Salvatore Mandrà. Establishing the quantum supremacy frontier with a 281 pflop/s simulation. *Quantum Science and Technology*, 2020.

[170] Andreas Walther, Frank Ziesel, Thomas Ruster, Sam T Dawkins, Konstantin Ott, Max Hettrich, Kilian Singer, Ferdinand Schmidt-Kaler, and Ulrich Poschinger. Controlling fast transport of cold trapped ions. *Physical Review Letters*, 109(8):080501, 2012.

[171] Ye Wang, Mark Um, Junhua Zhang, Shuoming An, Ming Lyu, Jing-Ning Zhang, L-M Duan, Dahyun Yum, and Kihwan Kim. Single-qubit quantum memory exceeding ten-minute coherence time. *Nature Photonics*, 11(10):646, 2017.

[172] Jack Wells, Buddy Bland, Jeff Nichols, Jim Hack, Fernanda Foertter, Gaute Hagen, Thomas Maier, Moetasim Ashfaq, Bronson Messer, and Suzanne Parete-Koon. Announcing supercomputer summit. 6 2016.

[173] Robert Wille, Majid Haghparast, Smaran Adarsh, and M Tanmay. Towards hdl-based synthesis of reversible circuits with no additional lines. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–7. IEEE, 2019.

[174] Robert Wille, Oliver Keszocze, Marcel Walter, Patrick Rohrs, Anupam Chattopadhyay, and Rolf Drechsler. Look-ahead schemes for nearest neighbor optimization of 1D and 2D quantum circuits. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 292–297. IEEE, 2016.

[175] Robert Wille, Aaron Lye, and Rolf Drechsler. Optimal SWAP gate insertion for nearest neighbor quantum circuits. In *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 489–494. IEEE, 2014.

[176] Stephen Wimperis. Broadband, narrowband, and passband composite pulses for use in advanced nmr experiments. *Journal of Magnetic Resonance, Series A*, 109(2):221–231, 1994.

[177] Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. *Commun. ACM*, 30(6):520–540, June 1987.

[178] K Wright, KM Beck, S Debnath, JM Amini, Y Nam, N Grzesiak, J-S Chen, NC Pisenti, M Chmielewski, C Collins, et al. Benchmarking an 11-qubit quantum computer. *arXiv preprint arXiv:1903.08181*, 2019.

[179] Xin-Chuan Wu, Dripto M. Debroy, Yongshan Ding, Jonathan M. Baker, Yuri Alexeev, Kenneth R. Brown, and Frederic T. Chong. Tilt: Achieving higher fidelity on a trapped-ion linear-tape quantum computing architecture, 2020.

[180] Xin-Chuan Wu, Yongshan Ding, Yunong Shi, Yuri Alexeev, Hal Finkel, Kibaek Kim, and Frederic T Chong. ILP-based scheduling for linear-tape model trapped-ion quantum computers, 2019.

[181] Yukai Wu, Sheng-Tao Wang, and L-M Duan. Noise analysis for high-fidelity quantum entangling gates in an anharmonic linear Paul trap. *Physical Review A*, 97(6):062325, 2018.

[182] Ed Younis, Koushik Sen, Katherine Yelick, and Costin Iancu. Qfast: Quantum synthesis using a hierarchical continuous circuit space. *arXiv preprint arXiv:2003.04462*, 2020.

[183] Jinfeng Zeng, Yufeng Wu, Jin-Guo Liu, Lei Wang, and Jiangping Hu. Learning and inference on generative adversarial quantum circuits. *Physical Review A*, 99(5):052306, 2019.

[184] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, May 1977.

[185] XiangYu Zou, Tao Lu, Wen Xia, Xuan Wang, Weizhe Zhang, Sheng Di, Dingwen Tao, and Franck Cappello. Accelerating relative-error bounded lossy compression for HPC datasets with precomputation-based mechanisms. In *Proceedings of the 35th International Conference on Massive Storage Systems and Technology (MSST19)*. IEEE, 2019.

[186] Alwin Zulehner, Philipp Niemann, Rolf Drechsler, and Robert Wille. Accuracy and compactness in decision diagrams for quantum computation. In *Design, Automation and Test in Europe*, 2019.

[187] Alwin Zulehner, Alexandru Paler, and Robert Wille. Efficient mapping of quantum circuits to the ibm qx architectures. In *2018 Design, Automation & Test in Europe Conference & Exhibition*, pages 1135–1138. IEEE, 2018.

[188] Alwin Zulehner and Robert Wille. Advanced simulation of quantum computations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

[189] Alwin Zulehner and Robert Wille. Matrix-vector vs. matrix-matrix multiplication: Potential in DD-based simulation of quantum computations. *Quantum*, 1(1):1–1, 2019.