THE UNIVERSITY OF CHICAGO

FULL-STACK, CROSS-LAYER OPTIMIZATIONS FOR QUANTUM COMPUTING

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
PRANAV GOKHALE

CHICAGO, ILLINOIS
DECEMBER 2020

I dedicate this dissertation to my friends and family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to first thank my advisor, Fred Chong. Every weekly meeting with Fred left me with intellectual inspiration and drive to complete research projects. Each of the papers comprising my PhD can be traced to a brainstorming session or discussion between Fred and me.

I am also grateful to Ali Javadi-Abhari and Margaret Martonosi who sparked and guided my research interest in quantum computing in college. If not for them, I would not have thought to pursue a PhD and I would not have been introduced to Fred.

I appreciate my research colleagues and co-authors: Casey, Eric, Hank, Jonathan, Kaiwen, Kate, Ken, Martin, Natalie, Nate, Olivia, Samantha, Shilin, Sophia, Swarnadeep, Rich, Teague, Thomas, Yongshan, Yunong, and Ziqi. One of the joys of quantum computing is the chance to work with researchers across many disciplines.

Most importantly, I would like to thank my friends and family. I credit friendships for making the past three years full of personal growth and enriching experiences. Finally, I'm grateful for the steadfast love and support of my parents, sister, and brother-in-law.

ABSTRACT

Quantum computing is a promising technology that unlocks new computational frontiers by leveraging the principles of quantum mechanics. After three decades of theoretical research, quantum computers have emerged in experimental and industrial settings in the past decade. Much of the current quantum system stack mirrors modern classical stacks: hardware and software are cleanly separated by layered abstraction barriers. However, we argue that near-term applications of quantum computing will require software that has a full-stack view spanning from the end-user application down to the underlying hardware. Our thesis is supported by six examples of software optimizations that cross between layers of the traditional system stack, thereby achieving gains that are otherwise invisible. The six optimizations include: (1) qutrit and higher-radix computing, (2) partial compilation of parametrized programs, (3) simultaneous measurement for Variational Quantum Eigensolver (VQE), (4) an asymptotic improvement in the runtime of VQE, (5) recompilation of quantum programs with pulse-augmented basis gates, and (6) speedups via quantum fan-out. We conclude by discussing future research directions that could enable practical speedups from near-term quantum computers.

CHAPTER 1
INTRODUCTION

As Moore's Law comes to an end, computing will require a paradigm shift for continued technological progress. Quantum computing provides such an opportunity, by leveraging the unique physics of quantum hardware to accelerate calculations beyond what is possible in the setting of classical physics. Properties such as superposition and entanglement enable a quantum computer with n qubits to simultaneously span a state space of size 2^n. Through clever processing techniques, this advantage can be turned into polynomial or even exponential sized speedups over classical algorithms for certain tasks. Famously, this includes speedups for tasks such as search [9] and factoring [10].

While quantum computing was originally proposed in the 1980s [11, 12], experimentally promising realizations did not emerge until the 2000s. For example, the transmon *qubit* (quantum bit)—a superconducting circuit that maintains a macroscopic quantum state—was introduced in 2007 [13]. Just 12 years later, the Google quantum team argued their computer with 53 transmon qubits solved a task in seconds [14] that would take days [15] on the world's most powerful classical supercomputer. In the meantime, qubits based on trapped ion, neutral atom, photonic, and silicon based technologies have also demonstrated significant potential in recent experiments.

In the longer term when systems with thousands or millions of qubits and low error rates become available, we will be able to build *logical qubits* that perform quantum error correction. However, in the near-term era of Noisy Intermediate-Scale Quantum (NISQ) [16] computing, we hope to find quantum speedups for practical applications without requiring error correction. This is a formidable challenge, but an exciting one that could accelerate the realization of practical value from quantum computing by several years, if not decades.

This dissertation examines the software stack for quantum computers in view of the challenge of obtaining quantum speedups in the near-term era. The core thesis statement is as follows:

> Optimizations cutting across the full stack are critical for computing advantage in the NISQ (Noisy Intermediate-Scale Quantum) era.

This thesis statement is best understood in contrast to the conventional wisdom, which advocates a stack that has clean layers, separated by well defined and rigid abstraction barriers. This quantum stack design is very analogous to classical computing, in that there are no leaky abstractions. Notably, this makes programming easier. For example, a modern application developer does not need to deal with the device physics of transistors or even low-level assembly code.

However, the rigidity of the abstraction barriers leads to efficiency losses. This is tolerable in classical computing, where we have the luxury of billions of bits, virtually perfect gates, and virtually infinite computation durations. But in the NISQ era of quantum computing, we have only a few dozen qubits, imperfect gates with $> 0.1\%$ typical errors, and only enough qubit coherence lifetime to support a few dozen gate layers. In this era, every optimization must be leveraged to extract as much mileage as possible from the underlying quantum hardware.

This motivates us to architect a system stack that allows for leaky abstractions. Taken to the extreme, a quantum application programmer should be able to interface all the way down to the underlying quantum hardware and perform optimizations that cut across the full stack. The right side of Figure 1.1 depicts the system architecture that we envision. While we recognize the value of conceptually separate layers, we also explicitly advocate for links or leaky abstractions across the entire stack to enable cross-layer optimizations. For completeness, we explicitly define each layer of the stack:

- Application. Here we consider any end-user application, in a form that can be expressed independently of a quantum (or classical) algorithm. For example, we might consider optimization applications that arise in power grids (hence the icon of power lines).

- Algorithm. Here we consider the wide range of quantum algorithms, expressed in a mathematical form.
- Compiler. This is software that translates the algorithm down to a form that is closer to the underlying hardware. The compiler generally acts on a quantum circuit description of the algorithm and outputs control pulses.
- Control. This is the classical hardware used to emit signals to the qubits. We focus on Arbitrary Waveform Generators (AWGs) which are depicted in the icon.
- Hardware. This is where we finally encounter the actual qubits, which could be implemented by many underlying technologies such as superconducting circuits, trapped ions, neutral atoms, or photonics.

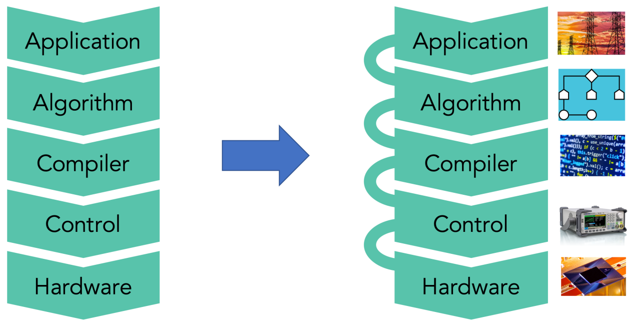


Figure 1.1: In this dissertation, we argue that the traditional quantum system stack on the left should be re-architected to allow for leaky abstractions, as on the right. Doing so will enable optimizations that cross layers and cut across the full stack. These optimizations are critical for computing advantage in the NISQ (Noisy Intermediate-Scale Quantum) era.

This dissertation comprises six papers that each introduce a new type of optimization cutting across the full stack, in alignment with the thesis statement. Each chapter aims to be self-contained. We begin in Chapter 2 with Asymptotic Improvements to Quantum Circuits via Qutrits [17], which demonstrates how quantum algorithms can be accelerated by leveraging higher-level states of underlying hardware. In particular, we note that the binary qu*bit* abstraction is simply a truncation of a quantum system to the bottom two energy levels.

We show that by instead accessing three-level qu*trit* hardware, we can compile a quantum algorithm (Generalized Toffoli) more efficiently. Relative to a qubit-only circuit, we attain an exponential speedup in the ancilla-free regime. We also perform simulations demonstrating the expected advantage of our technique.

Chapter 3, Partial Compilation of Variational Algorithms for Noisy Intermediate-Scale Quantum Machines [18], studies the compilation of a new paradigm of quantum algorithm. In this new *variational* paradigm, a sequence of quantum circuits is generated at runtime. This complicates compilation since compiler latency is now runtime latency, unlike other quantum algorithms which are statically pre-compiled. Our work demonstrates how variational algorithms can be partially compiled into optimized controls that target the underlying hardware.

Chapter 4, Minimizing State Preparations in Variational Quantum Eigensolver by Partitioning into Commuting Families [19], presents a technique for leveraging simultaneous measurement in the VQE algorithm. As a result, we substantially reduce the repetition cost of this algorithm. We also develop software for generating simultaneous measurement circuits, and we study the statistics of simultaneous measurement.

Chapter 5, $O(N^3)$ Measurement Cost for Variational Quantum Eigensolver on Molecular Hamiltonians [20], extends our work on reducing the repetition cost of VQE. We demonstrate that simultaneous measurement actually leads to an asymptotic reduction of this cost from $O(N^4)$ to $O(N^3)$, and we develop a tool for optimally orchestrating these simultaneous measurements.

Chapter 6, Optimized Quantum Compilation for Near-Term Algorithms with OpenPulse [21], introduces a concept known as an augmented basis gate. Unlike a standard basis gate in quantum assembly, an augmented basis gate is better aligned with the native physical interactions executed by the underlying qubit hardware. We compile algorithms in view of augmented basis gates in order to generate optimized control pulses. Through the

OpenPulse interface for pulse-level control, we experimentally demonstrate significant fidelity improvements on IBM hardware due to our technique.

Chapter 7, Optimizations with Fan-out in Near-Term Quantum Computers [22], evaluates the ability of quantum hardware to simultaneously execute gates on overlapping qubits. We demonstrate that both trapped ion and superconducting qubits have this capability, and we use it to improve the compilation of a wide range of near-term quantum algorithms. Our technique is validated with trapped ion simulation as well as experimental results on superconducting qubits.

Finally, we conclude in Chapter 8 with closing discussion and suggestions for future work. For completeness, Appendix A presents the author's curriculum vitae.

CHAPTER 2
ASYMPTOTIC IMPROVEMENTS TO QUANTUM CIRCUITS VIA QUTRITS

2.1 Introduction

Recent advances in both hardware and software for quantum computation have demonstrated significant progress towards practical outcomes. In the coming years, we expect quantum computing will have important applications in fields ranging from machine learning and optimization [23] to drug discovery [24]. While early research efforts focused on longer-term systems employing full error correction to execute large instances of algorithms like Shor factoring [10] and Grover search [9], recent work has focused on NISQ (Noisy Intermediate Scale Quantum) computation [16]. The NISQ regime considers near-term machines with just tens to hundreds of quantum bits (qubits) and moderate errors.

Given the severe constraints on quantum resources, it is critical to fully optimize the compilation of a quantum algorithm in order to have successful computation. Prior architectural research has explored techniques such as mapping, scheduling, and parallelism [25, 26, 27] to extend the amount of useful computation possible. In this work, we consider another technique: quantum trits (qutrits).

While quantum computation is typically expressed as a two-level binary abstraction of qubits, the underlying physics of quantum systems are not intrinsically binary. Whereas classical computers operate in binary states at the physical level (e.g. clipping above and below a threshold voltage), quantum computers have natural access to an infinite spectrum of discrete energy levels. In fact, hardware must actively suppress higher level states in order to achieve the two-level qubit approximation. Hence, using three-level qutrits is simply a choice of including an additional discrete energy level, albeit at the cost of more opportunities for error.

Prior work on qutrits (or more generally, d-level *qudits*) identified only constant factor gains from extending beyond qubits. In general, this prior work [28] has emphasized the information compression advantages of qutrits. For example, N qubits can be expressed as $\frac{N}{\log_2(3)}$ qutrits, which leads to $\log_2(3) \approx 1.6$-constant factor improvements in runtimes.

Our approach utilizes qutrits in a novel fashion, essentially using the third state as temporary storage, but at the cost of higher per-operation error rates. Under this treatment, the runtime (i.e. circuit depth or critical path) is *asymptotically* faster, and the reliability of computations is also improved. Moreover, our approach only applies qutrit operations in an intermediary stage: the input and output are still qubits, which is important for initialization and measurement on real devices [29, 30].

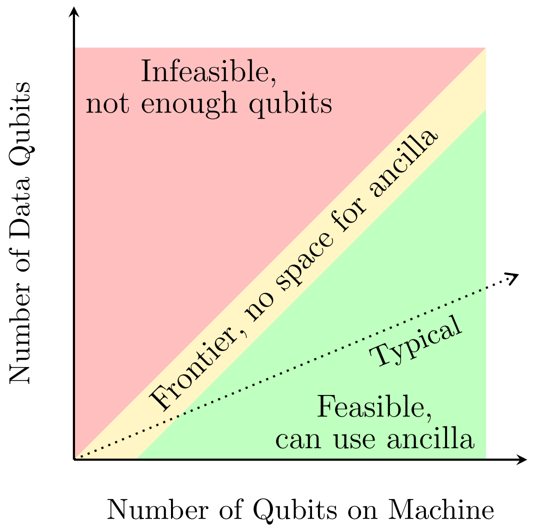


Figure 2.1: The frontier of what quantum hardware can execute is the yellow region adjacent to the 45° line. In this region, each machine qubit is a data qubit. Typical circuits rely on non-data ancilla qubits for workspace and therefore operate below the frontier.

The net result of our work is to extend the frontier of what quantum computers can compute. In particular, the frontier is defined by the zone in which every machine qubit is a data qubit, for example a 100-qubit algorithm running on a 100-qubit machine. This is indicated by the yellow region in Figure 2.1. In this frontier zone, we do not have room for non-data workspace qubits known as ancilla. The lack of ancilla in the frontier zone is a costly constraint that generally leads to inefficient circuits. For this reason, typical circuits instead operate below the frontier zone, with many machine qubits used as ancilla. Our work demonstrates that ancilla can be substituted with qutrits, enabling us to operate efficiently within the ancilla-free frontier zone.

We highlight the three primary contributions of our work:

1. A circuit construction based on qutrits that leads to asymptotically faster circuits ($633N \to 38 \log_2 N$) than equivalent qubit-only constructions. We also reduce total gate counts from $397N$ to $6N$.

2. An open-source simulator, based on Google's Cirq [31], which supports realistic noise simulation for qutrit (and qudit) circuits.

3. Simulation results, under realistic noise models, which demonstrate our circuit construction outperforms equivalent qubit circuits in terms of error. For our benchmarked circuits, our reliability advantage ranges from 2x for trapped ion noise models up to more than 10,000x for superconducting noise models. For completeness, we also benchmark our circuit against a qubit-only construction augmented by an ancilla and find our construction is still more reliable.

The rest of this chapter is organized as follows: Section 2.2 presents relevant background about quantum computation and Section 2.3 outlines related prior work that we benchmark our work against. Section 2.4 demonstrates our key circuit construction, and Section 2.5 surveys applications of this construction toward important quantum algorithms. Section 2.6 introduces our open-source qudit circuit simulator. Section 2.7 explains our noise modeling methodology (with full details in Section 2.10), and Section 2.8 presents simulation results for these noise models. Finally, we discuss our results at a higher level in Section 2.9.

2.2 Background

A qubit is the fundamental unit of quantum computation. Compared to their classical counterparts which take values of either 0 and 1, qubits may exist in a superposition of the two states. We designate these two basis states as $|0\rangle$ and $|1\rangle$ and can represent any qubit as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ with $\|\alpha\|^2 + \|\beta\|^2 = 1$. $\|\alpha\|^2$ and $\|\beta\|^2$ correspond to the probabilities of measuring $|0\rangle$ and $|1\rangle$ respectively.

Quantum states can be acted on by quantum gates which (a) preserve valid probability distributions that sum to 1 and (b) guarantee reversibility. For example, the X gate transforms a state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ to $X|\psi\rangle = \beta|0\rangle + \alpha|1\rangle$. The X gate is also an example of a classical reversible operation, equivalent to the NOT operation. In quantum computation, we have a single irreversible operation called measurement that transforms a quantum state into one of the two basis states with a given probability based on α and β.

In order to interact different qubits, two-qubit operations are used. The CNOT gate appears both in classical reversible computation and in quantum computation. It has a control qubit and a target qubit. When the control qubit is in the $|1\rangle$ state, the CNOT performs a NOT operation on the target. The CNOT gate serves a special role in quantum computation, allowing quantum states to become entangled so that a pair of qubits cannot be described as two individual qubit states. Any operation may be conditioned on one or more controls.

Many classical operations, such as AND and OR gates, are irreversible and therefore cannot directly be executed as quantum gates. For example, consider the output of 1 from an OR gate with two inputs. With only this information about the output, the value of the inputs cannot be uniquely determined. These operations can be made reversible by the addition of extra, temporary workspace bits initialized to 0. Using a single additional ancilla, the AND operation can be computed reversibly as in Figure 2.2.

Physical systems in classical hardware are typically binary. However, in common quantum

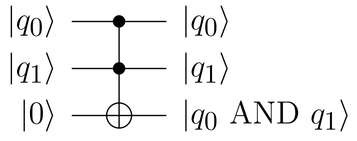


Figure 2.2: Reversible AND circuit using a single ancilla bit. The inputs are on the left, and time flows rightward to the outputs. This AND gate is implemented using a Toffoli (CCNOT) gate with inputs q_0, q_1 and a single ancilla initialized to 0. At the end of the circuit, q_0 and q_1 are preserved, and the ancilla bit is set to 1 if and only if both other inputs are 1.

hardware, such as in superconducting and trapped ion computers, there is an infinite spectrum of discrete energy levels. The qubit abstraction is an artificial approximation achieved by suppressing all but the lowest two energy levels. Instead, the hardware may be configured to manipulate the lowest three energy levels by operating on qutrits. In general, such a computer could be configured to operate on any number of d levels, except as d increases the number of opportunities for error, termed error channels, increases. Here, we focus on $d = 3$ with which we achieve the desired improvements to the Generalized Toffoli gate.

In a three level system, we consider the computational basis states $|0\rangle$, $|1\rangle$, and $|2\rangle$ for qutrits. A qutrit state $|\psi\rangle$ may be represented analogously to a qubit as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle + \gamma|2\rangle$, where $\|\alpha\|^2 + \|\beta\|^2 + \|\gamma\|^2 = 1$. Qutrits are manipulated in a similar manner to qubits; however, there are additional gates which may be performed on qutrits.

For instance, in quantum binary logic, there is only a single X gate. In ternary, there are three X gates denoted X_{01}, X_{02}, and X_{12}. Each of these X_{ij} for $i \neq j$ can be viewed as swapping $|i\rangle$ with $|j\rangle$ and leaving the third basis element unchanged. For example, for a qutrit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle + \gamma|2\rangle$, applying X_{02} produces $X_{02}|\psi\rangle = \gamma|0\rangle + \beta|1\rangle + \alpha|2\rangle$. Each of these operations' actions can be found in the left state diagram in Figure 2.3.

There are two additional non-trivial operations on a single trit. They are the $+1$ and -1 (sometimes referred to as a $+2$) operations (with $+$ meaning addition modulo 3). These operations can be written as $X_{01}X_{12}$ and $X_{12}X_{01}$, respectively; however, for simplicity, we will refer to them as X_{+1} and X_{-1} operations. A summary of these gates' actions can be found in the right state diagram in Figure 2.3.

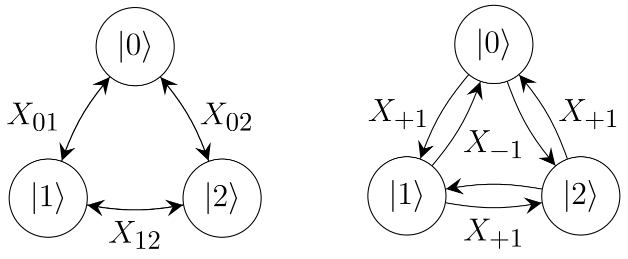


Figure 2.3: The five nontrivial permutations on the basis elements for a qutrit. (Left) Each operation here switches two basis elements while leaving the third unchanged. These operations are self-inverses. (Right) These two operations permute the three basis elements by performing a $+1 \mod 3$ and $-1 \mod 3$ operation. They are each other's inverses.

Other, non-classical, operations may be performed on a single qutrit. For example, the Hadamard gate [32] can be extended to work on qutrits in a similar fashion as the X gate was extended. In fact, all single qubit gates, like rotations, may be extended to operate on qutrits. In order to distinguish qubit and qutrit gates, all qutrit gates will appear with an appropriate subscript.

Just as single qubit gates have qutrit analogs, the same holds for two qutrit gates. For example, consider the CNOT operation, where an X gate is performed conditioned on the control being in the $|1\rangle$ state. For qutrits, any of the X gates presented above may be performed, conditioned on the control being in any of the three possible basis states. Just as qubit gates are extended to take multiple controls, qutrit gates are extended similarly. The set of single qutrit gates, augmented by any entangling two-qutrit gate, is sufficient for universality in ternary quantum computation [33].

One question concerning the feasibility of using higher states beyond the standard two is whether these gates can be implemented and perform the desired manipulations. Qutrit gates have been successfully implemented [1, 34, 35] indicating it is possible to consider higher level systems apart from qubit only systems.

In order to evaluate a decomposition of a quantum circuit, we consider quantum circuit costs. The space cost of a circuit, i.e. the number of qubits (or qutrits), is referred to as circuit *width*. Requiring ancilla increases the circuit width and therefore the space cost of a circuit. The time cost for a circuit is the *depth* of a circuit. The depth is given as the length of the critical path (in terms of gates) from input to output.

2.3 Prior Work

2.3.1 Qudits

Qutrits, and more generally qudits, have been been studied in past work both experimentally and theoretically. Experimentally, d as large as 10 has been achieved (including with two-qudit operations) [36], and $d = 3$ qutrits are commonly used internally in many quantum systems [37, 38].

However, in past work, qudits have conferred only an information compression advantage. For example, N qubits can be compressed to $\frac{N}{\log_2(d)}$ qudits, giving only a constant-factor advantage [28] at the cost of greater errors from operating qudits instead of qubits. Under the assumption of linear cost scaling with respect to d, it has been demonstrated that $d = 3$ is optimal [39, 40], although as we show in Section 2.7 the cost is generally superlinear in d.

The information compression advantage of qudits has been applied specifically to Grover's search algorithm [41, 42, 43, 44] and to Shor's factoring algorithm [45]. Ultimately, the tradeoff between information compression and higher per-qudit errors has not been favorable in past work. As such, the past research towards building practical quantum computers has focused on qubits.

Our work introduces qutrit-based circuits which are *asymptotically* better than equivalent qubit-only circuits. Unlike prior work, we demonstrate a compelling advantage in both runtime and reliability, thus justifying the use of qutrits.

	This Work	Gidney [4]	He [46]	Barenco [3]	Wang [43]	Lanyon [47], Ralph [48]
Depth	$\log N$	N	$\log N$	N^2	N	N
Ancilla	0	0	N	0	0	0
Qudit Types	Controls are qutrits	Qubits	Qubits	Qubits	Controls are qutrits	Target is $d = N$-level qudit
Constants	Small	Large	Small	Small	Small	Small

Table 2.1: Asymptotic comparison of N-controlled gate decompositions. The total gate count for all circuits scales linearly (except for Barenco [3], which scales quadratically). Our construction uses qutrits to achieve logarithmic depth without ancilla. We benchmark our circuit construction against Gidney [4], which is the asymptotically best ancilla-free qubit circuit.

2.3.2 Generalized Toffoli Gate

We focus on the Generalized Toffoli gate, which simply adds more controls to the Toffoli circuit in Figure 2.2. The Generalized Toffoli gate is an important primitive used across a wide range of quantum algorithms, and it has been the focus of extensive past optimization work. Table 2.1 compares past circuit constructions for the Generalized Toffoli gate to our construction, which is presented in full in Section 2.4.2.

Among prior work, the Gidney [4], He [46], and Barenco [3] designs are all qubit-only. The three circuits have varying tradeoffs. While Gidney and Barenco operate at the ancilla-free frontier, they have large circuit depths: linear with a large constant for Gidney and quadratic for Barenco. The Gidney design also requires rotation gates for very small angles, which poses an experimental challenge. While the He circuit achieves logarithmic depth, it requires an ancilla for each data qubit, effectively halving the effective potential of any given quantum hardware. Nonetheless, in practice, most circuit implementations use these linear-ancilla constructions due to their small depths and gate counts.

As in our approach, circuit constructions from Lanyon [47], Ralph [48], and Wang [43] have attempted to improve the ancilla-free Generalized Toffoli gate by using qudits. Both the Lanyon [47] and Ralph [48] constructions, which have been demonstrated experimentally, achieve linear circuit depths by operating the target as a $d = N$-level qudit. Wang [43] also achieves a linear circuit depth but by operating each control as a qutrit.

Our circuit construction, presented in Section 2.4.2, has similar structure to the He design, which can be represented as a binary tree of gates. However, instead of storing temporary results with a linear number of ancilla qubits, our circuit temporarily stores information directly in the qutrit $|2\rangle$ state of the controls. Thus, no ancilla are needed.

In our simulations, we benchmark our circuit construction against the Gidney construction [4] because it is the asymptotically best qubit circuit in the ancilla-free frontier zone. We label these two benchmarks as QUTRIT and QUBIT. The QUBIT circuit handles the lack of ancilla by using *dirty* ancilla, which unlike *clean* (initialized to $|0\rangle$) ancilla, can have an unknown initial state. Dirty ancilla can therefore be bootstrapped internally from a quantum circuit. However, this technique requires a large number of Toffoli gates which makes the decomposition particularly expensive in gate count.

Augmenting the base Gidney construction with a single ancilla[1] does reduce the constants for the decomposition significantly, although the asymptotic depth and gate counts are maintained. For completeness, we also benchmark our circuit against this augmented construction, QUBIT+ANCILLA. However, the augmented circuit does not operate at the ancilla-free frontier, and it conflicts with parallelism, as discussed in Section 2.9.

2.4 Circuit Construction

In order for quantum circuits to be executable on hardware, they are typically decomposed into single- and two- qudit gates. Performing efficient low depth and low gate count decompositions is important in both the NISQ regime and beyond. Our circuits assume all-to-all connectivity–we discuss this assumption in Section 2.9.

2.4.1 Key Intuition

To develop intuition for our technique, we first present a Toffoli gate decomposition which lays the foundation for our generalization to multiple controls. In each of the following

1. This ancilla can also also be dirty.

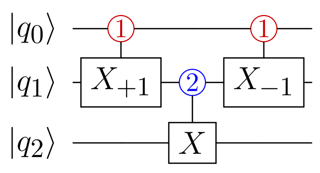


Figure 2.4: A Toffoli decomposition via qutrits. Each input and output is a qubit. The red controls activate on $|1\rangle$ and the blue controls activate on $|2\rangle$. The first gate temporarily elevates q_1 to $|2\rangle$ if both q_0 and q_1 were $|1\rangle$. We then perform the X operation only if q_1 is $|2\rangle$. The final gate restores q_0 and q_1 to their original state.

constructions, all inputs and outputs are qubits, but we may occupy the $|2\rangle$ state temporarily during computation. Maintaining binary input and output allows these circuit constructions to be inserted into any preexisting qubit-only circuits.

In Figure 2.4, a Toffoli decomposition using qutrits is given. A similar construction for the Toffoli gate is known from past work [47, 48]. The goal is to perform an X operation on the last (target) input qubit q_2 if and only if the two control qubits, q_0 and q_1, are both $|1\rangle$. First a $|1\rangle$-controlled X_{+1} is performed on q_0 and q_1. This elevates q_1 to $|2\rangle$ iff q_0 and q_1 were both $|1\rangle$. Then a $|2\rangle$-controlled X gate is applied to q_2. Therefore, X is performed only when both q_0 and q_1 were $|1\rangle$, as desired. The controls are restored to their original states by a $|1\rangle$-controlled X_{-1} gate, which undoes the effect of the first gate. The key intuition in this decomposition is that the qutrit $|2\rangle$ state can be used instead of ancilla to store temporary information.

2.4.2 Generalized Toffoli Gate

We now present our circuit decomposition for the Generalized Toffoli gate in Figure 2.5. The decomposition is expressed in terms of three-qutrit gates (two controls, one target) instead of single- and two- qutrit gates, because the circuit can be understood purely classically at this granularity. In actual implementation and in our simulation, we used a decomposition [1] that requires 6 two-qutrit and 7 single-qutrit physically implementable quantum gates.

Our circuit decomposition is most intuitively understood by treating the left half of the

the circuit as a tree. The desired property is that the root of the tree, q_7, is $|2\rangle$ if and only if each of the 15 controls was originally in the $|1\rangle$ state. To verify this property, we observe the root q_7 can only become $|2\rangle$ iff q_7 was originally $|1\rangle$ and q_3 and q_{11} were both previously $|2\rangle$. At the next level of the tree, we see q_3 could have only been $|2\rangle$ if q_3 was originally $|1\rangle$ and both q_1 and q_5 were previously $|2\rangle$, and similarly for the other triplets. At the bottom level of the tree, the triplets are controlled on the $|1\rangle$ state, which are only activated when the even-index controls are all $|1\rangle$. Thus, if any of the controls were not $|1\rangle$, the $|2\rangle$ states would fail to propagate to the root of the tree. The right half of the circuit performs *uncomputation* to restore the controls to their original state.

After each subsequent level of the tree structure, the number of qubits under consideration is reduced by a factor of ~ 2. Thus, the circuit depth is logarithmic in N. Moreover, each qutrit is operated on by a constant number of gates, so the total number of gates is linear in N.

Our circuit decomposition still works in a straightforward fashion when the control type of the top qubit, q_0, activates on $|2\rangle$ or $|0\rangle$ instead of activating on $|1\rangle$. These two constructions are necessary for the Incrementer circuit in 2.5.3.

We verified our circuits, both formally and via simulation. Our verification scripts are available on our GitHub [49].

2.5 Application to Algorithms

The Generalized Toffoli gate is an important primitive in a broad range of quantum algorithms. In this section, we survey some of the applications of our circuit decomposition.

2.5.1 Artificial Quantum Neuron

The artificial quantum neuron [50] is a promising target application for our circuit construction, because the algorithm's circuit implementation is dominated by large Generalized Toffoli gates. The algorithm may exhibit an exponential advantage over classical perceptron encoding and it has already been executed on current quantum hardware. Moreover, the threshold behavior of perceptrons has inherent noise resilience, which makes the artificial quantum neuron particularly promising as a near-term application on noisy systems. The current implementation of the neuron on IBM quantum computers relies on ancilla qubits [51] which constrains the circuit width to $N = 4$ data qubits. Our circuit construction offers a path to larger circuit sizes without waiting for larger hardware.

2.5.2 Grover's Algorithm

Grover's Algorithm for search over M unordered items requires just $O(\sqrt{M})$ oracle queries. However, each oracle query is followed by a post-processing step which requires a multiply-controlled gate with $N = \lceil \log_2 M \rceil$ controls [32]. The explicit circuit diagram is shown in Figure 2.6.

Our log-depth circuit construction directly applies to the multiply-controlled gate. Thus, we reduce a $\log M$ factor in Grover search's time complexity to $\log \log M$ via our ancilla-free qutrit decomposition.

2.5.3 Incrementer

The Incrementer circuit performs the $+1 \mod 2^N$ operation to a register of N qubits. While logarithmic circuit depth can be achieved with linear ancilla qubits [52], the best ancilla-free incrementers require either linear depth with large linearity constants [53] or quadratic depth [3]. Using alternate control activations for our Generalized Toffoli gate decomposition, the incrementer circuit is reduced to $O(\log^2 N)$ depth with no ancilla, a significant improvement over past work.

Our incrementer circuit construction is shown in Figure 2.7 for an $N = 8$ wide register. The multiple-controlled X_{+1} gates perform the job of computing carries: a carry is performed iff the least significant bit generates (represented by the $|2\rangle$ control) and all subsequent bits propagate (represented by the consecutive $|1\rangle$ controls). We present an $N = 8$ incrementer here and have verified the general construction, both by formal proof and by explicit circuit simulation for larger N.

The critical path of this circuit is the chain of $\log N$ multiply-controlled gates (of width $\frac{N}{2}$, $\frac{N}{4}$, $\frac{N}{8}$, ...) which act on $|a_0\rangle$. Since our multiply-controlled gate decomposition has log-depth, we arrive at a total circuit depth circuit scaling of $\log^2 N$.

2.5.4 Arithmetic Circuits and Shor's Algorithm

The Incrementer circuit is a key subcircuit in many other arithmetic circuits such as constant addition, modular multiplication, and modular exponentiation. Further, the modular exponentiation circuit is the bottleneck in the runtime for executing Shor's algorithm for factorization [53, 54]. While a shallower Incrementer circuit alone is not sufficient to reduce the asymptotic cost of modular exponentiation (and therefore Shor's algorithm), it does reduce constants relative to qubit-only circuits.

2.5.5 Error Correction and Fault Tolerance

The Generalized Toffoli gate has applications to circuits for both error correction [55] and fault tolerance [56]. We foresee two paths of applying these circuits. First, our circuit construction can be used to construct error-resilient *logical qubits* more efficiently. This is critical for quantum algorithms like Grover's and Shor's which are expected to require such logical qubits. In the nearer-term, NISQ algorithms are likely to make use of limited error correction. For instance, recent results have demonstrated that error correcting a single qubit at a time for the Variational Quantum Eigensolver algorithm can significantly reduce total error [57]. Thus, our circuit construction is also relevant for NISQ-era error correction.

2.6 Simulator

To simulate our circuit constructions, we developed a qudit simulation library, built on Google's Cirq Python library [31]. Cirq is a qubit-based quantum circuit library and includes a number of useful abstractions for quantum states, gates, circuits, and scheduling.

Our work extends Cirq by discarding the assumption of two-level qubit states. Instead, all state vectors and gate matrices are expanded to apply to d-level qudits, where d is a circuit parameter. We include a library of common gates for $d = 3$ qutrits. Our software adds a comprehensive noise simulator, detailed below in Section 2.6.1.

In order to verify our circuits are logically correct, we first simulated them with noise disabled. We extended Cirq to allow gates to specify their action on classical non-superposition input states without considering full state vectors. Therefore, each classical input state can be verified in space and time proportional to the circuit width. By contrast, Cirq's default simulation procedure relies on a dense state vector representation requiring space and time exponential in the circuit width. Reducing this scaling from exponential to linear dramatically improved our verification procedure, allowing us to verify circuit constructions for all possible classical inputs across circuit sizes up to widths of 14.

Our software is fully open source [49].

2.6.1 Noise Simulation

Figure 2.8 depicts a schematic view of our noise simulation procedure which accounts for both gate errors and idle errors, described below. To determine when to apply each gate and idle error, we use Cirq's scheduler which schedules each gate as early as possible, creating a sequence of `Moment`'s of simultaneous gates. During each `Moment`, our noise simulator applies a gate error to every qudit acted on. Finally, the simulator applies an idle error to every qudit. This noise simulation methodology is consistent with previous simulation techniques which have accounted for either gate errors [58] or idle errors [59].

Gate errors arise from the imperfect application of quantum gates. Two-qudit gates are noisier than single-qudit gates [5], so we apply different noise channels for the two. Our specific gate error probabilities are given in Section 2.7.

Idle errors arise from the continuous decoherence of a quantum system due to energy relaxation and interaction with the environment. The idle errors differ from gate errors in two ways which require special treatment:

1. Idle errors depend on duration, which in turn depend on the schedule of simultaneous gates (`Moments`). In particular, two-qudit gates take longer to apply than single-qudit gates. Thus, if a `Moment` contains a two-qudit gate, the idling errors must be scaled appropriately. Our specific scaling factors are given in Section 2.7.

2. For the generic model of gate errors, the error channel is applied with probability independent of the quantum state. This is not true for idle errors such as T_1 amplitude damping, which only applies when the qudit is in an excited state. This is treated in the simulator by computing idle error probabilities during each `Moment`, for each qutrit.

Gate errors are reduced by performing fewer *total gates*, and idle errors are reduced by decreasing the circuit *depth*. Since our circuit constructions asymptotically decrease the depth, this means our circuit constructions scale favorably in terms of asymptotically fewer idle errors.

Our full noise simulation procedure is summarized in Algorithm 1. The ultimate metric of interest is the mean *fidelity*, which is defined as the squared overlap between the ideal (noise-free) and actual output state vectors. Fidelity expresses the probability of overall successful execution. We do not consider initialization errors and readout errors, because our circuit constructions maintain binary input and output, only occupying the qutrit $|2\rangle$ states during intermediate computation. Therefore, the initialization and readout errors for our circuits are identical to those for conventional qubit circuits.

Algorithm 1: Pseudocode for each simulation trial, given a particular circuit and noise model.

```
$|\Psi\rangle \leftarrow$ random initial state vector
$|\Psi\rangle_{\text{ideal}}$ = circuit applied to $|\Psi\rangle$ without noise
foreach Moment do
    foreach Gate $\in$ Moment do
        $|\psi\rangle \leftarrow$ Gate applied to $|\psi\rangle$
        GateError $\leftarrow$ DrawRand(GateError Prob.)
        $|\psi\rangle \leftarrow$ GateError applied to $|\psi\rangle$
    end
    foreach Qutrit do
        if Moment has 2-qudit gate then
            IdleErrors $\leftarrow$ long-duration idle errors
        else
            IdleErrors $\leftarrow$ short-duration idle errors
        end
        Prob. $\leftarrow [\|M|\Psi\rangle\|^2$ for M $\in$ IdleErrors]
        IdleError $\leftarrow$ DrawRand(Prob.)
        $|\psi\rangle \leftarrow$ IdleError applied to $|\psi\rangle$
        Renormalize($|\psi\rangle$)
    end
end
return $\langle\Psi_{\text{ideal}}|\Psi\rangle^2$, fidelity between ideal & actual output;
```

We also do not consider crosstalk errors, which occur when gates are executed in parallel. The effect of crosstalk is very device-dependent and difficult to generalize. Moreover, crosstalk can be mitigated by breaking each `Moment` into a small number of sub-moments and then scheduling two-qutrit operations to reduce crosstalk, as demonstrated in prior work [60, 61].

2.6.2 Simulator Efficiency

Simulating a quantum circuit with a classical computer is, in general, exponentially difficult in the size of the input because the state of N qudits is represented by a state vector of d^N complex numbers. For 14 qutrits, with complex numbers stored as two 8-byte floats (`complex128` in NumPy), a state vector occupies 77 megabytes.

A naive circuit simulation implementation would treat every quantum gate or `Moment` as a $d^N \times d^N$ matrix. For 14 qutrits, a single such matrix would occupy 366 terabytes–out of range of simulability. While the exponential nature of simulating our circuits is unavoidable, we mitigate the cost by using a variety of techniques which rely only on state vectors, rather than full square matrices. For example, we maintain Cirq's approach of applying gates by Einstein Summation [62], which obviates computation of the $d^N \times d^N$ matrix corresponding to every gate or `Moment`.

Our noise simulator only relies on state vectors, by adopting the quantum trajectory methodology [63, 64], which is also used by the Rigetti PyQuil noise simulator [65]. At a high level, the effect of noise channels like gate and idle errors is to turn a coherent quantum state into an incoherent mix of classical probability-weighted quantum states (for example, $|0\rangle$ and $|1\rangle$ with 50% probability each). The most complete description of such an incoherent quantum state is called the density matrix and has dimension $d^N \times d^N$. The quantum trajectory methodology is a stochastic approach–instead of maintaining a density matrix, only a single state is propagated and the error term is drawn randomly at each timestep. Over repeated trials, the quantum trajectory methodology converges to the same results as from full density matrix simulation [65]. Our simulator employs this technique–each simulation in Algorithm 1 constitutes a single quantum trajectory trial. At every step, a specific `GateError` or `IdleError` term is picked, based on a weighted random draw.

Finally, our random state vector generation function was also implemented in $O(d^N)$ space and time. This is an improvement over other open source libraries [66, 67], which perform random state vector generation by generating full $d^N \times d^N$ unitary matrices from a Haar-random distribution and then truncating to a single column. Our simulator directly computes the first column and circumvents the full matrix computation.

With optimizations, our simulator is able to simulate circuits up to 14 qutrits in width. This is in the range as other state-of-the-art noisy quantum circuit simulations [68] (since 14 qutrits $\approx$ 22 qubits). While each simulation trial took several minutes (depending on the particular circuit and noise model), we were able to run trials in parallel over multiple processes and multiple machines, as described in Section 2.8.

2.7 Noise Models

In this section, we describe our noise models at a high level, with mathematical details described in Section 2.10. We chose noise models which represent realistic near-term machines. We first present a generic, parametrized noise model roughly applicable to all quantum systems. We then present specific parameters, under the generic noise model, which apply to near-term superconducting quantum computers. Finally, we present a specific noise model for trapped ion quantum computers.

2.7.1 Generic Noise Model

Gate Errors

The scaling of gate errors for a d-level qudit can be roughly summarized as increasing as d^4 for two-qudit gates and d^2 for single-qudit gates. For $d = 2$, there are 4 single-qubit gate error channels and 16 two-qubit gate error channels. For $d = 3$ there are 9 and 81 single- and two- qutrit gate error channels respectively. Consistent with other simulators [65, 59], we use the symmetric depolarizing gate error model, which assumes equal probabilities between each error channel. Under these noise models, two-qutrit gates are $(1 - 80p_2)/(1 - 15p_2)$ times less reliable than two-qubit gates, where p_2 is the probability of each two-qubit gate error channel. Similarly, single-qutrit gates are $(1 - 8p_1)/(1 - 3p_1)$ times less reliable than single-qubit gates, where p_1 is the probability of each single-qubit gate error channel.

Idle Errors

Our treatment of idle errors focuses on the relaxation from higher to lower energy states in quantum devices. This is called amplitude damping or T_1 relaxation. This noise channel irreversibly takes qudits to lower states. For qubits, the only amplitude damping channel is from $|1\rangle$ to $|0\rangle$, and we denote this damping probability as λ_1. For qutrits, we also model damping from $|2\rangle$ to $|0\rangle$, which occurs with probability λ_2.

2.7.2 Superconducting QC

We chose four noise models based on superconducting quantum computers expected in the next few years. These noise models comply with the generic noise model above and are thus parametrized by p_1, p_2, λ_1 and λ_2. The λ_i probabilities are derived from two other experimental parameters: the gate time Δt and T_1, a timescale that captures how long a qudit persists coherently.

As a starting point for representative near-term noise models, we consider parameters for *current* superconducting quantum computers. For IBM's public cloud-accessible superconducting quantum computers, we have $3p_1 \approx 10^{-3}$ and $15p_2 \approx 10^{-2}$. The duration of single- and two- qubit gates is $\Delta t \approx 100ns$ and $\Delta t \approx 300ns$ respectively, and the IBM devices have $T_1 \approx 100\mu s$ [5, 6].

However, simulation for these current parameters indicates an error is almost certain to occur during execution of a modest size 14-input Generalized Toffoli circuit. This motivates us to instead consider noise models for better devices which are a few years away. Accordingly, we adopt a baseline superconducting noise model, labeled as SC, corresponding to a superconducting device which has 10x lower gate errors and 10x longer T_1 duration than the current IBM hardware. This range of parameters has already been achieved experimentally in superconducting devices for gate errors [69, 70] and for T_1 duration [71, 72] independently. Faster gates (shorter Δt) are yet another path towards greater noise resilience. We do not vary gate speeds, because errors only depend on the $\Delta t/T_1$ ratio, and we already vary T_1. In practice however, faster gates could also improve noise-resilience.

We also consider three additional near-term device noise models, indexed to the SC noise model. These three models further improve gate errors, T_1, or both, by a 10x factor. The specific parameters are given in Table 2.2. Our 10x improvement projections are realistic extrapolations of progress in hardware. In particular, Schoelkopf's Law–the quantum analogue of Moore's Law–has observed that T_1 durations have increased by 10x every 3 years for the past 20 years [73]. Hence, 100x longer T_1 is a reasonable projection for devices that are ~ 6 years away.

2.7.3 Trapped Ion $^{171}Yb^{+}$ QC

We also simulated noise models for trapped ion quantum computing devices. Trapped ion devices are well matched to our qutrit-based circuit constructions because they feature all-to-

Noise Model	$3p_1$	$15p_2$	T_1
SC	10^{-4}	10^{-3}	1 ms
SC+T1	10^{-4}	10^{-3}	10 ms
SC+GATES	10^{-5}	10^{-4}	1 ms
SC+T1+GATES	10^{-5}	10^{-4}	10 ms

Table 2.2: Noise models simulated for superconducting devices. Current publicly accessible IBM superconducting quantum computers have single- and two- qubit gate errors of $3p_1 \approx 10^{-3}$ and $15p_2 \approx 10^{-2}$, as well as T_1 lifetimes of 0.1 ms [5, 6]. Our baseline benchmark, SC, assumes 10x better gate errors and T_1. The other three benchmarks add a further 10x improvement to T_1, gate errors, or both.

all connectivity [74], and many ions that are ideal candidates for QC devices are naturally multi-level systems.

We focus on the $^{171}\text{Yb}^+$ ion, which has been experimentally demonstrated as both a qubit and qutrit [29, 30]. Trapped ions are often favored in QC schemes due to their long T_1 times. One of the main advantages of using a trapped ion is the ability to take advantage of magnetically insensitive states known as "clock states." By defining the computational subspace on these clock states, idle errors caused from fluctuations in the magnetic field are minimized–this is termed a DRESSED_QUTRIT, in contrast with a BARE_QUTRIT. However, compared to superconducting devices, gates are much slower. Thus, gate errors are the dominant error source for ion trap devices. We modelled a fundamental source of these errors: the spontaneous scattering of photons originating from the lasers used to drive the gates. The duration of single- and two- qubit gates used in this calculation was $\Delta t \approx 1$ μs and $\Delta t \approx 200$ μs respectively [75]. The single- and two- qudit gate error probabilities are given in Table 2.3.

2.8 Results

Figure 2.9 plots the exact circuit depths for all three benchmarked circuits. The qubit-based circuit constructions from past work are linear in depth and have a high linearity constant.

Noise Model	p_1	p_2
TI_QUBIT	6.4×10^{-4}	1.3×10^{-4}
BARE_QUTRIT	2.2×10^{-4}	4.3×10^{-4}
DRESSED_QUTRIT	1.5×10^{-4}	3.1×10^{-4}

Table 2.3: Noise models simulated for trapped ion devices. The single- and two- qutrit gate error channel probabilities are based on calculations from experimental parameters. For all three models, we use single- and two- qudit gate times of $\Delta t \approx 1\ \mu s$ and $\Delta t \approx 200\ \mu s$ respectively.

Augmenting with a single borrowed ancilla reduces the circuit depth by a factor of 8. However, both circuit constructions are surpassed significantly by our qutrit construction, which scales logarithmically in N and has a relatively small leading coefficient.

Figure 2.10 plots the total number of two-qudit gates for all three circuit constructions. As noted in Section 2.4, our circuit construction is not asymptotically better in total gate count–all three plots have linear scaling. However, as emphasized by the logarithmic vertical axis, the linearity constant for our qutrit circuit is 70x smaller than for the equivalent ancilla-free qubit circuit and 8x smaller than for the borrowed-ancilla qubit circuit.

Our simulations under realistic noise models were run in parallel on over 100 n1-standard-4 Google Cloud instances. These simulations represent over 20,000 CPU hours, which was sufficient to estimate mean fidelity to an error of $2\sigma < 0.1\%$ for each circuit-noise model pair.

The full results of our circuit simulations are shown in Figure 2.11. All simulations are for the 14-input (13 controls, 1 target) Generalized Toffoli gate. We simulated each of the three circuit benchmarks against each of our noise models (when applicable), yielding the 16 bars in the figure.

2.9 Discussion

Figure 2.11 demonstrates that our QUTRIT construction (orange bars) significantly outperforms the ancilla-free QUBIT benchmark (blue bars) in fidelity (success probability) by more than 10,000x.

For the SC, SC+T1, and SC+GATES noise models, our qutrit constructions achieve between 57-83% mean fidelity, whereas the ancilla-free qubit constructions all have almost 0% fidelity. Only the lowest-error model, SC+T1+GATES achieves modest fidelity of 26% for the QUBIT circuit, but in this regime, the qutrit circuit is close to 100% fidelity.

The trapped ion noise models achieve similar results–the DRESSED_QUTRIT and the BARE_QUTRIT achieve approximately 95% fidelity via the QUTRIT circuit, whereas the TI_QUBIT noise model has only 45% fidelity. Between the dressed and bare qutrits, the dressed qutrit exhibits higher fidelity than the bare qutrit, as expected. Moreover, as discussed in Section 2.10.3, the dressed qutrit is resilient to leakage errors, so the simulation results should be viewed as a lower bound on its advantage over the qubit and bare qutrit.

Based on these results, trapped ion qutrits are a particularly strong match to our qutrit circuits. In addition to attaining the highest fidelities, trapped ions generally have all-to-all connectivity [74] within each ion chain, which is critical as our circuit construction requires operations between distant qutrits.

The superconducting noise models also achieve good fidelities. They exhibit a particularly large advantage over ancilla-free qubit constructions because idle errors are significant for superconducting systems, and our qutrit construction significantly reduces idling (circuit depth). However, most superconducting quantum systems only feature nearest-neighbor or short-range connectivity. Accounting for data movement on a nearest-neighbor-connectivity 2D architecture would expand the qutrit circuit depth from $\log N$ to $\sqrt{N}$ (since the distance between any two qutrits would scale as $\sqrt{N}$). However, recent work has experimentally demonstrated fully-connected superconducting quantum systems via random access memory [76]. Such systems would also be well matched to our circuit construction.

For completeness, Figure 2.11 also shows fidelities for the QUBIT+ANCILLA circuit benchmark, which augments the ancilla-free QUBIT circuit with a single dirty ancilla. Since QUBIT+ANCILLA has linearity constants $\sim$ 10x better than the ancilla-free qubit circuit, it exhibits significantly better fidelities. While our QUTRIT circuit still outperforms the QUBIT+ANCILLA circuit, we expect a crossing point where augmenting a qubit-only Generalized Toffoli with enough ancilla would eventually outperform QUTRIT. However, we emphasize that the gap between an ancilla-free and constant-ancilla construction for the Generalized Toffoli is actually a fundamental rather than an incremental gap, because:

- Constant-ancilla constructions prevent circuit parallelization. For example, consider the parallel execution of N/k disjoint Generalized Toffoli gates, each of width k for some constant k. An ancilla-free Generalized Toffoli would pose no issues, but an ancilla-augmented Generalized Toffoli would require $\Theta(N/k)$ ancilla. Thus, constant-ancilla constructions can impose a choice between serializing to linear depth or regressing to linear ancilla count. The Incrementer circuit in Figure 2.7 is a concrete example of this scenario–any multiply-controlled gate decomposition requiring a single clean ancilla or more than 1 dirty ancilla would contradict the parallelism and reduce runtime.

- Even if we only consider serial circuits, given the exponential advantage of certain quantum algorithms, there is a significant practical difference between operating at the ancilla-free frontier and operating just a few data qubits below the frontier.

While we only performed simulations up to 14 inputs in width, we would see an even bigger advantage in larger circuits because our construction has asymptotically lower depth and therefore asymptotically lower idle errors. We also expect to see an advantage for the circuits in Section 2.5 that rely on the Generalized Toffoli, although we did not explicitly simulate these circuits.

Our circuit construction and simulation results point towards promising directions of future work that we highlight below:

- A number of useful quantum circuits, especially arithmetic circuits, make extensive use of multiply-controlled gates. However, these circuits are typically pre-compiled into single- and two- qubit gates using one of the decompositions from prior work, usually one that involves ancilla qubits. Revisiting these arithmetic circuits from first principles, with our qutrit circuit as a new tool, could yield novel and improved circuits like our Incrementer circuit in Section 2.5.3.

- Relatedly, we see value in a logic synthesis tool that injects qutrit optimizations into qubit circuits, automated in fashion inspired by classical reversible logical synthesis tools [77, 78].

- While $d = 3$ qutrits were sufficient to achieve the desired asymptotic speedups for our circuits of interest, there may be other circuits that are optimized by qudit information carriers for larger d. In particular, we note that increasing d and thereby increasing information compression may be advantageous for hardware with limited connectivity.

Independent of these future directions, the results presented in this work are applicable to quantum computing in the near term, on machines that are expected within the next five years. The net result of this work is to extend the frontier of what is computable by quantum hardware, and hence to accelerate the timeline for practical quantum computing, rather than waiting for better hardware. Emphatically, our results are driven by the use of qutrits for *asymptotically* faster ancilla-free circuits. Moreover, we also improve linearity constants by two orders of magnitudes. Finally, as verified by our open-source circuit simulator coupled with realistic noise models, our circuits are more reliable than qubit-only equivalents. Our results justify the use of qutrits as a path towards scaling quantum computers.

2.10 Detailed Noise Model

We chose noise models that represent realistic near-term machines. We first present a generic, parametrized noise model in that is roughly applicable to all quantum systems. Next, we present specific parameters, under the generic noise model, that apply to near-term superconducting quantum computers. Finally, we present a specific noise model for $^{171}\mathrm{Yb}^{+}$ trapped ions.

2.10.1 Generic Noise Model

The general form of a quantum noise model is expressed by the Kraus Operator formalism which specifies a set of matrices, $\{K_i\}$, each capturing an error channel. Under this formalism, the evolution of a system with initial state $\sigma = |\Psi\rangle\langle\Psi|$ is expressed as a function $\mathcal{E}(\sigma)$, where:

$$\mathcal{E}(\sigma) = \mathcal{E}\left(|\Psi\rangle\langle\Psi|\right) = \sum_i K_i \sigma K_i^{\dagger} \tag{2.1}$$

where † denotes the matrix conjugate-transpose.

Gate Errors

For a single qubit, there are four possible error channels: no-error, bit flip, phase flip, and phase+bit flip. These channels can be expressed as products of the Pauli matrices:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \text{and} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

which correspond to bit and phase flips respectively. The no-error channel is $X^0 Z^0 = I$ and the phase+bit flip channel is the product $X^1 Z^1$.

In the Kraus operator formalism, we express this single-qubit gate error model as

$$\mathcal{E}(\sigma) = \sum_{j=0}^{1}\sum_{k=0}^{1} p_{jk}(X^j Z^k)\sigma(X^j Z^k)^\dagger \tag{2.2}$$

where p_{jk} denotes the probability of the corresponding Kraus operator.

This gate error model is called the Pauli or depolarizing channel. We assume all error terms have equal probabilities, i.e. $p_{jk} = p_1$ for $j, k \neq 0$. This assumption of symmetric depolarizing is standard and is used by most noise simulators [59]. Under this model, the error channel simplifies to:

$$\mathcal{E}(\sigma) = (1 - 3p_1)\sigma + \sum_{jk \in \{0,1\}^2 \setminus 00} p_1(X^j Z^k)\sigma(X^j Z^k)^\dagger \tag{2.3}$$

For two-qubit gate errors, the Kraus operators are the Cartesian product of the two single-qubit gate error Kraus operators, leading to the noise channel:

$$\mathcal{E}(\sigma) = (1 - 15p_2)\sigma + \sum_{jklm \in \{0,1\}^4 \setminus 0000} p_2 K_{jklm}\sigma K_{jklm}^\dagger \tag{2.4}$$

where p_2 is the probability of each error term and $K_{jklm} = X^j Z^k \otimes X^l Z^m$.

Next, for qutrits, we have a similar form, except that there are now more possible error channels. We now use the generalized Pauli matrices:

$$X_{+1} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad \text{and} \quad Z_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & e^{2\pi i/3} & 0 \\ 0 & 0 & e^{4\pi i/3} \end{pmatrix}$$

The Cartesian product of $\{I, X_{+1}, X_{+1}^2\}$ and $\{I, Z_3, Z_3^2\}$ constitutes a basis for all 3x3 matrices. Hence, this Cartesian product also constitutes the Kraus operators for the single-qutrit gate error [79, 80, 58]:

$$\mathcal{E}(\sigma) = (1 - 8p_1)\sigma + \sum_{jk \in \{0,1,2\}^2 \backslash 00} p_1 (X_{+1}^j Z_3^k)\sigma(X_{+1}^j Z_3^k)^\dagger \tag{2.5}$$

and similarly, the two-qutrit gate error channel is:

$$\mathcal{E}(\sigma) = (1 - 80p_2)\sigma + \sum_{\substack{jklm \in \\ \{0,1,2\}^4 \backslash 0000}} p_2 K_{jklm} \sigma K_{jklm}^\dagger \tag{2.6}$$

Note that in this model, the dominant effect of using qutrits instead of qubits is that the no-error probability for two-operand gates diminishes from $1 - 15p_2$ to $1 - 80p_2$, as expressed by equations 2.4 and 2.6 respectively.

Idle Errors

For qubits, the Kraus operators for amplitude damping are:

$$K_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\lambda_1} \end{pmatrix} \quad \text{and} \quad K_1 = \begin{pmatrix} 0 & \sqrt{\lambda_1} \\ 0 & 0 \end{pmatrix} \tag{2.7}$$

For qutrits, the Kraus operator for amplitude damping can be modeled as [80, 81]:

$$K_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \sqrt{1-\lambda_1} & 0 \\ 0 & 0 & \sqrt{1-\lambda_2} \end{pmatrix}, K_1 = \begin{pmatrix} 0 & \sqrt{\lambda_1} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

$$\text{and } K_2 = \begin{pmatrix} 0 & 0 & \sqrt{\lambda_2} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{2.8}$$

As discussed in Section 2.6.1, these noise channels are incoherent (non-unitary), which means that the probability of each error occurring depends on the current state. Specifically, the probability of the K_i channel affecting the state $|\Psi\rangle$ is $\| K_i |\psi\rangle \|^2$ [32].

2.10.2 Superconducting QC

We picked four noise models based on superconducting quantum computers that are expected in the next few years. These noise models comply with the generic noise model above and are thus parametrized by p_1, p_2, λ_1, and λ_2. The λ_m terms are given by [81]:

$$\lambda_m = 1 - e^{-m\Delta t/T_1} \tag{2.9}$$

where Δt is the duration of the idling and T_1 is associated with the lifetime of each qubit.

2.10.3 Trapped Ion $^{171}Yb^{+}$ QC

Based on calculations from experimental parameters for the trapped ion qutrit, we know the specific Kraus operator types for the error terms, which deviate slightly from those in the generic error model. The specific Kraus operator matrices are provided at our GitHub repository [49].

We chose three noise models: TI_QUBIT, BARE_QUTRIT, and DRESSED_QUTRIT. Both TI_QUBIT and DRESSED_QUTRIT take advantage of clock states and thus have very small idle errors. They both would be ideal candidates for a qudit. The BARE_QUTRIT will suffer more from idle errors as it is not strictly defined on clock states but will require less experimental resources to prepare. Idle errors are very small in magnitude and manifest as coherent phase errors rather than amplitude damping errors as modeled in Section 2.7.1. We also do not consider leakage errors. These errors could be handled for Yb^+ by treating each ion as a $d = 4$ qudit, regardless of whether we use it as a qubit or a qutrit.

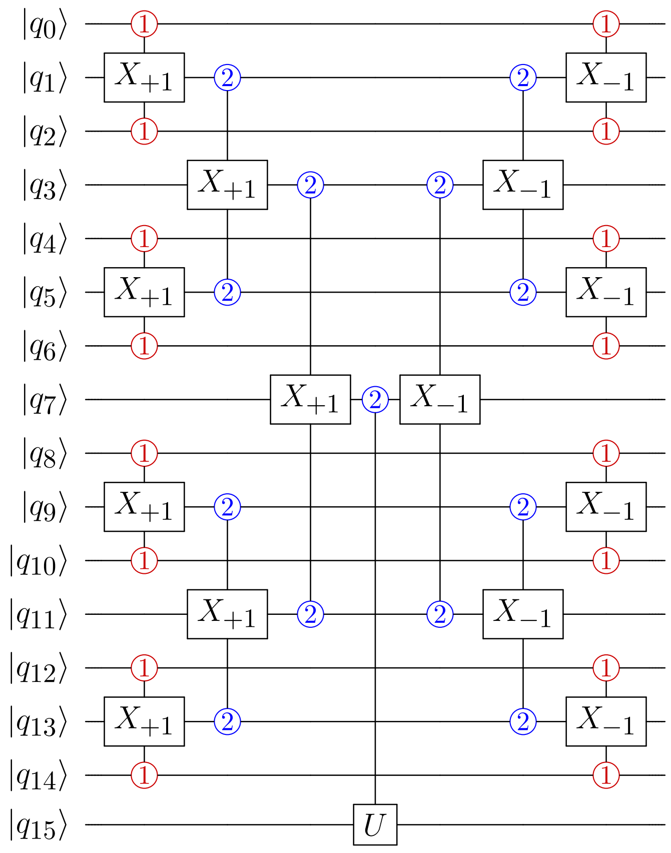


Figure 2.5: Our circuit decomposition for the Generalized Toffoli gate is shown for 15 controls and 1 target. The inputs and outputs are both qubits, but we allow occupation of the $|2\rangle$ qutrit state in between. The circuit has a tree structure and maintains the property that the root of each subtree can only be elevated to $|2\rangle$ if all of its control leaves were $|1\rangle$. Thus, the U gate is only executed if all controls are $|1\rangle$. The right half of the circuit performs uncomputation to restore the controls to their original state. This construction applies more generally to any multiply-controlled U gate. Note that the three-input gates are decomposed into 6 two-input and 7 single-input gates in our actual simulation, as based on the decomposition in [1].

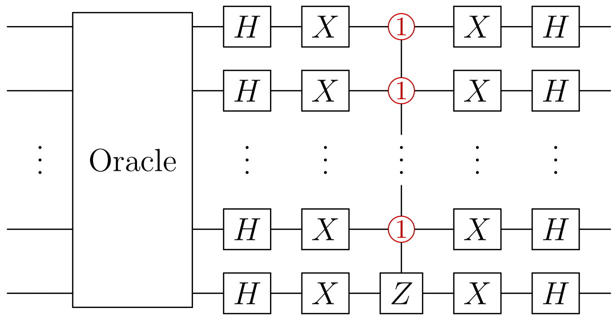


Figure 2.6: Each iteration of Grover Search has a multiply-controlled Z gate. Our logarithmic depth decomposition, reduces a $\log M$ factor in Grover's algorithm to $\log\log M$.

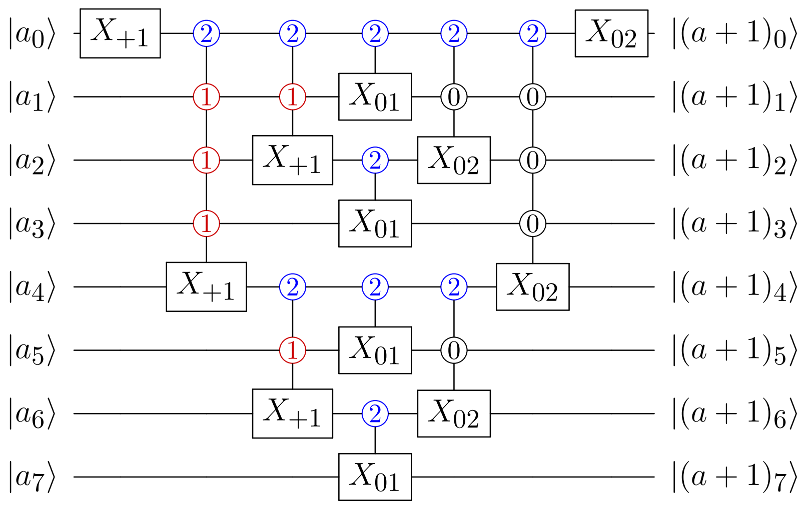


Figure 2.7: Our circuit decomposition for the Incrementer. At each subcircuit in the recursive design, multiply-controlled gates are used to efficiently propagate carries over half of the subcircuit. The $|2\rangle$ control checks for carry generation and the chain of $|1\rangle$ controls checks for carry propagation. The circuit depth is $\log^2 N$, which is only possible because of our log depth multiply-controlled gate primitive.

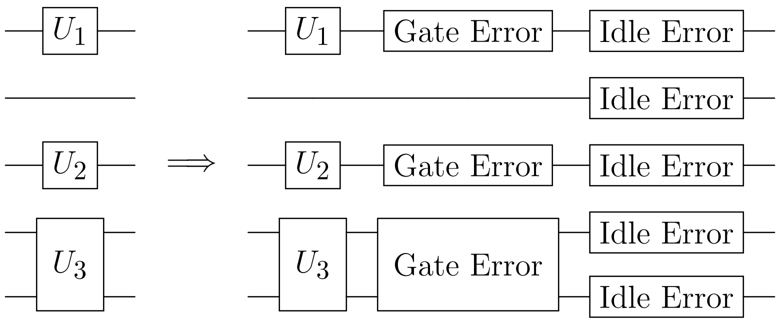


Figure 2.8: This `Moment` comprises three gates executed in parallel. To simulate with noise, we first apply the ideal gates, followed by a gate error noise channel on each affected qudit. This gate error noise channel depends on whether the corresponding gate was single- or two-qudit. Finally, we apply an idle error to every qudit. The idle error noise channel depends on the duration of the `Moment`.

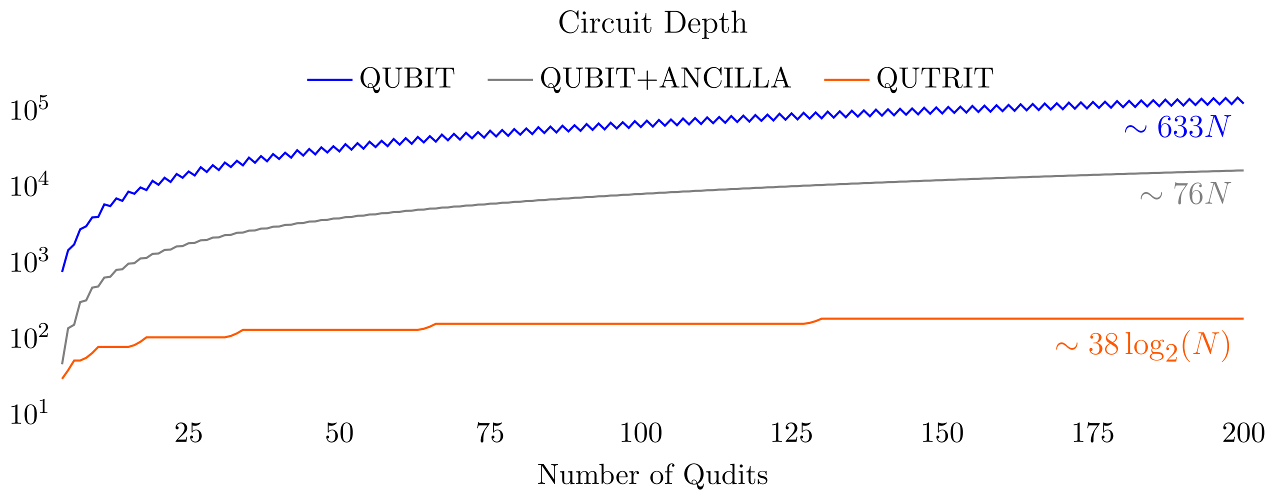


Figure 2.9: Exact circuit depths for all three benchmarked circuit constructions for the N-controlled Generalized Toffoli up to $N = 200$. Both QUBIT and QUBIT+ANCILLA scale linearly in depth and both are bested by QUTRIT's logarithmic depth.

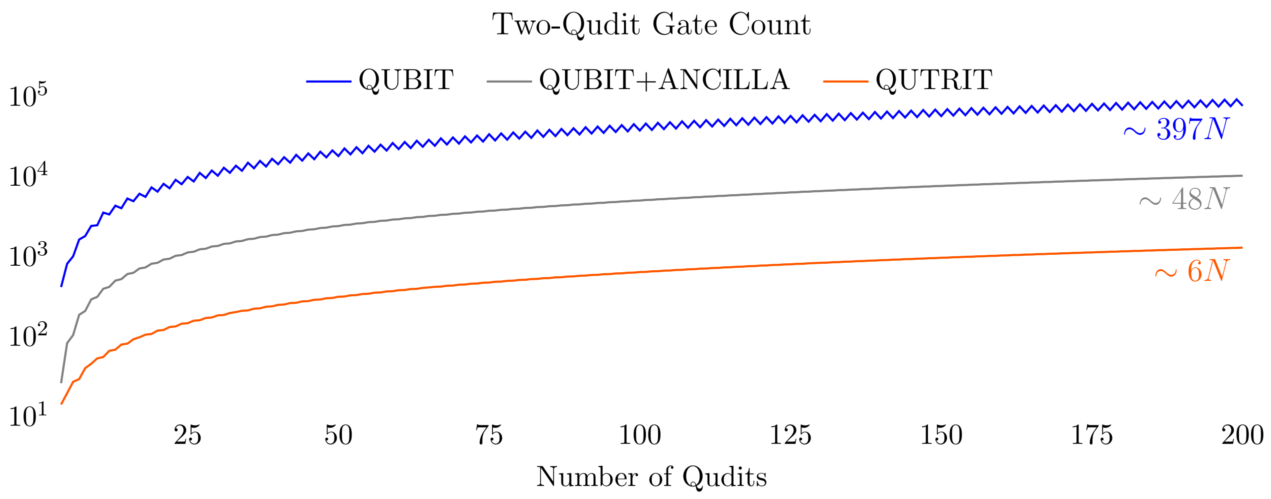


Figure 2.10: Exact two-qudit gate counts for the three benchmarked circuit constructions for the N-controlled Generalized Toffoli. All three plots scale linearly; however the QUTRIT construction has a substantially lower linearity constant.

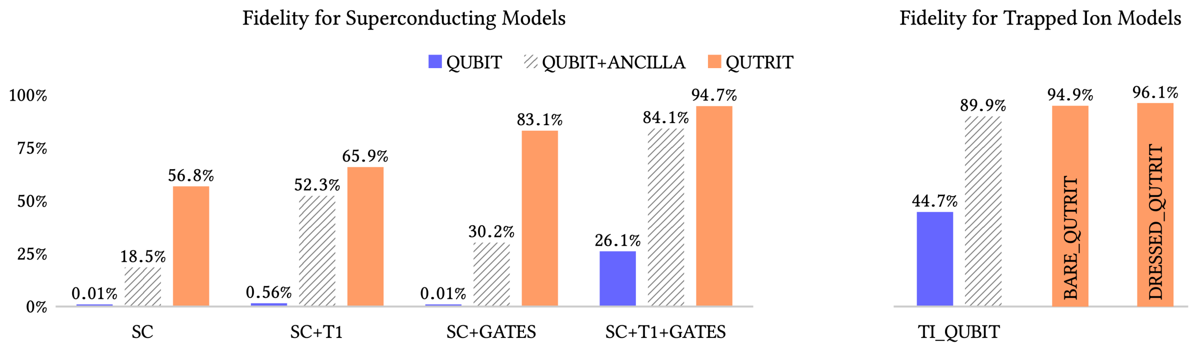


Figure 2.11: Circuit simulation results for all possible pairs of circuit constructions and noise models. Each bar represents 1000+ trials, so the error bars are all $2\sigma < 0.1\%$. Our QUTRIT construction significantly outperforms the QUBIT construction. The QUBIT+ANCILLA bars are drawn with dashed lines to emphasize that it has access to an extra ancilla bit, unlike our construction.

CHAPTER 3
PARTIAL COMPILATION OF VARIATIONAL ALGORITHMS FOR NOISY INTERMEDIATE-SCALE QUANTUM MACHINES

3.1 Introduction

In the Noisy Intermediate-Scale Quantum (NISQ) era, we expect to operate hardware with hundreds or thousands of quantum bits (qubits), acted on by imperfect gates [16]. Moreover, connectivity in these NISQ machines will be sparse and qubits will have modest lifetimes. Given these limitations, NISQ era machines will not be able to execute large-scale quantum algorithms like Shor Factoring [10] and Grover Search [9], which rely on error correction that requires millions of qubits [82, 83].

However, recently, *variational algorithms* have been introduced that are well matched to NISQ machines. This new class of algorithms has a wide range of applications such as molecular ground state estimation [84], MAXCUT approximation [85], and prime factorization [86]. The two defining features of a variational algorithm are that:

1. the algorithm complies with the constraints of NISQ hardware. Thus, the circuit for a variational algorithm should have modest requirements in qubit count (circuit width) and runtime (circuit depth / critical path).

2. the quantum circuit for the algorithm is parametrized by a list of angles. These parameters are optimized by a classical optimizer over the course of many iterations. For this reason, variational algorithms are also termed as hybrid quantum-classical algorithms [16]. Typically, a classical optimizer that is robust to small amounts of noise (e.g. Nelder-Mead) is chosen [84, 87].

Standard non-variational quantum algorithms are fully specified at compile time and therefore can be fully optimized by static compilation tools as in previous work [88, 89]. By contrast, each iteration of a variational algorithm depends on the results of the previous iteration–hence, compilation must be interleaved through the computation. As even small instances of variational algorithms will require thousands of iterations [90], the compilation latency for each iteration therefore becomes a serious limitation. This feature of variational algorithms is a significant departure from previous non-variational quantum algorithms.

To cope with this limitation on compilation latency, past work on variational algorithms has performed compilation under the standard gate-based model. This methodology has the advantage of extremely fast compilation–a lookup table maps each gate to a sequence of machine-level control pulses so that compilation simply amounts to concatenating the pulses corresponding to each gate. We note that this compilation procedure is a conservative picture of experimental approaches to gate-based compilation. In practice, parametrized gates may be handled by a step-function lookup table that depends on the run-time parameters, with the aim of reducing errors, as demonstrated in [91, 92, 93].

The gate-based compilation model is known to fall short of the GRadient Ascent Pulse Engineering (GRAPE) [94, 95] compilation technique, which compiles directly to the level of the machine-level control pulses that a quantum computer actually executes. In past work [96, 97, 98], GRAPE has been used to achieve 2-5x pulse speeedups over gate-based compilation for a range of quantum algorithms. Since fidelity decreases exponentially in time, with respect to the extremely short lifetimes of qubits, reducing the pulse duration is critical to ensuring that a computation completes before being completely scrambled by quantum decoherence effects. Thus, 2-5x pulse speedups translate to an even bigger advantage in the success probability of a quantum circuit.

However, GRAPE-based compilation has a substantial cost: compilation time. Running GRAPE control on a circuit with just four qubits takes several minutes. For representative four qubit circuits, we observed compile times ranging from 10 minutes to 1 hour, even with state-of-the-art hardware and GPU acceleration. This would amount to several weeks or months of total compilation latency over the course of thousands of iterations (and millions of iterations will be needed for larger problems). By contrast, typical pulse times for quantum circuits are on the order of microseconds, so the compilation latency imposed by GRAPE is untenable. Thus, GRAPE-based compilation is not practical out-of-the-box for variational algorithms.

In this chapter, we introduce *partial compilation*, a strategy that approaches the pulse duration speedup of GRAPE, but with a manageable overhead in compilation latency. With this powerful new compiler capability, **we enable the architectural choice of pulse-level instructions**, which supports more complex near-term applications through lower latencies and thus much lower error rates. This architectural choice would be infeasible without our compiler support. Our specific contributions include:

- Demonstration of the advantage of GRAPE over gate based compilation for variational algorithms

- Strict partial compilation, a strategy that pre-computes optimal pulses for parametrization-independent blocks of gates. This strategy is strictly better than gate-based compilation– it achieves a significant pulse speedup (approaching GRAPE results), with no overhead in compilation latency.

- Flexible partial compilation, a strategy that performs as well as full GRAPE, but with a dramatic speedup in compilation latency via precomputed hyperparameter optimization.

The rest of this chapter is organized as follows. Section 3.2 gives prerequisite background on quantum computation and Section 3.3 describes related work from prior research. Section 3.4 describes characteristics of our benchmark variational algorithms, with particular attention to the structural properties that our compilation strategies exploit. Section 3.5 explains the GRAPE compilation methodology. Sections 3.6 and 3.7 explain our partial compilation strategies and Section 3.8 discusses our results. We conclude in Section 3.9 and propose future work in Section 3.10. Section 3.11 presents the system Hamiltonian that we consider in GRAPE.

3.2 Background on Quantum Computation

3.2.1 Qubits

The fundamental unit of quantum computation is a quantum bit, or qubit. A qubit has two basis states, which are represented by *state vectors* denoted

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ and } |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Unlike a classical bit, the state of a qubit can be in a *superposition* of both $|0\rangle$ and $|1\rangle$. In particular, the space of valid qubit states are $\alpha|0\rangle + \beta|1\rangle$, normalized such that $|\alpha|^2 + |\beta|^2 = 1$. When a qubit is measured, its quantum state *collapses* and either $|0\rangle$ or $|1\rangle$ is measured, with probabilities $|\alpha|^2$ and $|\beta|^2$ respectively.

A two-qubit system has four basis states:

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, |10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \text{ and } |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

and any two-qubit state can be expressed as the superposition $\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$ (normalized so that $|\alpha| + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$). More generally, an N-qubit system has 2^N basis states. Therefore, 2^N numbers, called amplitudes, are needed to describe the state of a general N-qubit system. This exponential scaling gives rise to both the difficulty of classically simulating a quantum system, as well as the potential for quantum computers to exponentially outperform classical computers in certain applications.

3.2.2 Quantum Gates

A quantum algorithm is described in terms of a quantum circuit, which is a sequence of 1- and 2- input quantum gates. Every quantum gate is represented by a square matrix, and the action of a gate is to left-multiply a state vector by the gate's matrix. Because quantum states are normalized by measurement probabilities, these matrices must preserve l^2-norms. This corresponding set of matrices are *unitary* (orthogonal) matrices. The unitary matrices for two important single-qubit gates are:

$$R_x(\theta) = \begin{pmatrix} i\cos\frac{\theta}{2} & \sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & i\cos\frac{\theta}{2} \end{pmatrix} \text{ and } R_z(\phi) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$$

At $\theta = \pi$, the $R_x(\pi)$ gate has matrix $\left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, which acts as a NOT gate: left-multiplying by it swaps between the $|0\rangle$ and $|1\rangle$ states. This bit-flip gate is termed the X gate.

Similarly, at $\phi = \pi$, the $R_z(\pi)$ gate has matrix $\left(\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right)$, which applies a -1 multiplier to the amplitude of $|1\rangle$; this type of gate is unique to the quantum setting, where amplitudes can be negative (or complex). This 'phase'-flip gate is termed the Z gate.

An important 2-input quantum gate is

$$\text{CX} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

The CX gate, often referred to as the CNOT or Controlled-NOT gate, applies an action that is controlled on the first input. If the first input is $|0\rangle$, then the CX gate has no effect. If the first input is $|1\rangle$, then it applies an $X = R_x(\pi)$ to the second qubit.

The CX gate is an *entangling gate*, meaning that its effect cannot be decomposed into independent gates acting separately on the two qubits. An important result in quantum computation states that the set of all one qubit gates, plus a single entangling gate, is sufficient for universality [32]. Since the $R_x(\theta)$ and $R_z(\phi)$ gates span the set of all one qubit gates, we see that, $\{R_x(\theta), R_z(\phi), CX\}$ is a universal gate set.

In practice, we seek to implement a quantum algorithm using the most efficient quantum circuit possible, with efficiency defined in terms of circuit width (number of qubits) and depth (length of critical path, or runtime of the circuit). Accordingly, quantum circuits are optimized by repeatedly applying gate identities that reduce the resources consumed by the circuit. All circuits that are presented in this chapter were optimized using IBM Qiskit's Transpiler, which applies a variety of circuit identities–for example, aggressive cancellation of CX gates and 'Hadamard' gates. We also augmented the IBM optimizer with our own compiler pass for merging rotation gates–e.g. $R_x(\alpha)$ followed by $R_x(\beta)$ merges into $R_x(\alpha + \beta)$–which we found to further reduce circuit sizes.

3.2.3 Gate-Based Compilation

At the lowest level of hardware, quantum computers are controlled by analog pulses. Therefore, quantum compilation must translate from a high level quantum algorithm down to a sequence of control pulses. Once a quantum algorithm has been decomposed into a quantum circuit comprising single- and two- qubit gates, gate-based compilation simply proceeds by concatenating a sequence of pulses corresponding to each gate. In particular, a lookup table maps from each gate in the gate set to a sequence of control pulses that executes that gate. Table 3.1 indicates the total pulse duration for each gate in the compilation basis gate

Gate	$R_z(\phi)$	$R_x(\theta)$	H	CX	SWAP
Time (ns)	0.4	2.5	1.4	3.8	7.4

Table 3.1: Library of the compiler's gate set and corresponding pulse durations (in nanoseconds) for each gate. The runtimes of circuits under gate-based compilation are indexed to these pulse durations.

set. These pulse durations are based on the gmon-qubit [99] quantum system described in Section 3.11.

As previously noted, the $\{R_x(\theta), R_z(\phi), \text{CX}\}$ gate set alone is sufficient for universality, so in principle the H and SWAP gates could be removed from the compilation basis gate set. However, we include the generated pulses (using GRAPE as described below) for these gates in our compilation set, because quantum assembly languages typically include them in their basis set [100, 101, 102, 65, 103, 104].

The advantage of the gate-based approach is its short pulse compilation time, as the lookup and concatenation of pulses can be accomplished almost instantaneously. However, it prevents the optimization of pulses from happening across the gates, because there might exists a global pulse for the entire circuit that is shorter and more accurate than the concatenated one. The quality of the concatenated pulse relies heavily on an efficient gate decomposition of the quantum algorithm.

3.2.4 GRAPE

GRadient Pulse Engineering (GRAPE) is a strategy for compilation that numerically finds the best control pulses needed to execute a quantum circuit or subcircuit by following a gradient descent procedure [95, 105]. We use the Tensorflow implementation of GRAPE described in [97]. In contrast to the gate based approach, GRAPE does not have the limitation incurred by the gate decomposition. Instead, it directly searches for the optimal control pulse for the input circuit as a whole. Our full GRAPE procedure is described further detail in Section 3.5.

3.3 Related Work

Past publications of variational algorithm implementations have relied on gate-based compilation, using parametrized gates such as $R_x(\theta)$ and $R_z(\phi)$. Existing quantum languages offer support for such parametrized gates [104, 100, 101, 106, 65, 103]. In most languages, the angles must be declared at compile time–thus at every iteration of a variational algorithm, a new circuit is compiled based on the new parametrization. Rigetti's Quil [65] language goes a step further by supporting runtime resolution of the parameters in parameters gates, which allows dynamic implementations of variational algorithms. However, as acknowledged in the Quil specifications, this approach hampers circuit optimization, because the actual parameters are not known until runtime.

While this chapter treats gate-based compilation as a simple lookup table between gates and pulses, experimental implementations have already moved directionally towards GRAPE-style, because pulse sequences can depend on the input angles in a complicated fashion. For example, in [92], a parametrized $U(\phi)$ gate has five different pulse sequence decompositions, each corresponding to ϕ in ranges set by the breakpoints $[-\pi, 2.25, -0.25, 0.25, 2.25, \pi]$. [91] and [93] have similar step-function gate-to-pulse translation.

The growing overhead of compilation latency has been recognized, and recent work has proposed the development of specialized FPGAs for the compilation of variational algorithms [107]. More broadly, we note that pulse level control is at the cusp of industry adoption. An open specification for pulse-level control, OpenPulse, was standardized recently [100], and IBM plans to introduce an API for pulse level control in 2019 [108]. Pulse access to quantum machines will open the door to experimental realizations of GRAPE, including for variational algorithms as proposed in this chapter.

3.4 Variational Benchmarks

Variational quantum algorithms are important in the near-term because they comply with the constraints of NISQ hardware. In particular, variational algorithms have innate error resilience, due to the hybrid alternation with a noise-robust classical optimizer [84, 87]. Every iteration of a variational algorithm is parameterized by a list of angles. In general, the parameter space explored by a variational algorithm is not known a priori–the classical optimizer picks the next iteration's parameters based on the results of the previous iterations. Consequently, the compilation for each iteration is interleaved with the actual computation. A schematic of this process is illustrated in Figure 3.1.

There are two variational quantum algorithms: Variational Quantum Eigensolver and Quantum Approximate Optimization Algorithm. We discuss both below.

3.4.1 Variational Quantum Eigensolver

The Variational Quantum Eigensolver (VQE) is used to find the ground state energy of a molecule, a task that is exponentially difficult in general for a classical computer, but is believed to be efficiently solvable by a quantum computer [109]. Estimating the molecular ground state has important applications to chemistry such as determining reaction rates [110] and molecular geometry [111]. A conventional quantum algorithm for solving this problem is called the Quantum Phase Estimation (QPE) algorithm [112]. However, for a target precision ϵ, QPE requires a quantum circuit with depth $O(1/\epsilon)$, whereas VQE algorithm requires $O(1/\epsilon^2)$ iterations of depth-$O(1)$ circuits[113]. The latter assumes a much more relaxed fidelity requirement on the qubits and gate operations, because the higher the circuit depth, the more likely the circuit experiences an error at the end. At a high level, VQE can be conceptualized as a guess-check-repeat algorithm. The check stage involves the preparation of a quantum state corresponding to the guess. This preparation stage is done in polynomial time on a quantum computer, but would incur exponential cost (owing to the 2^N state vector

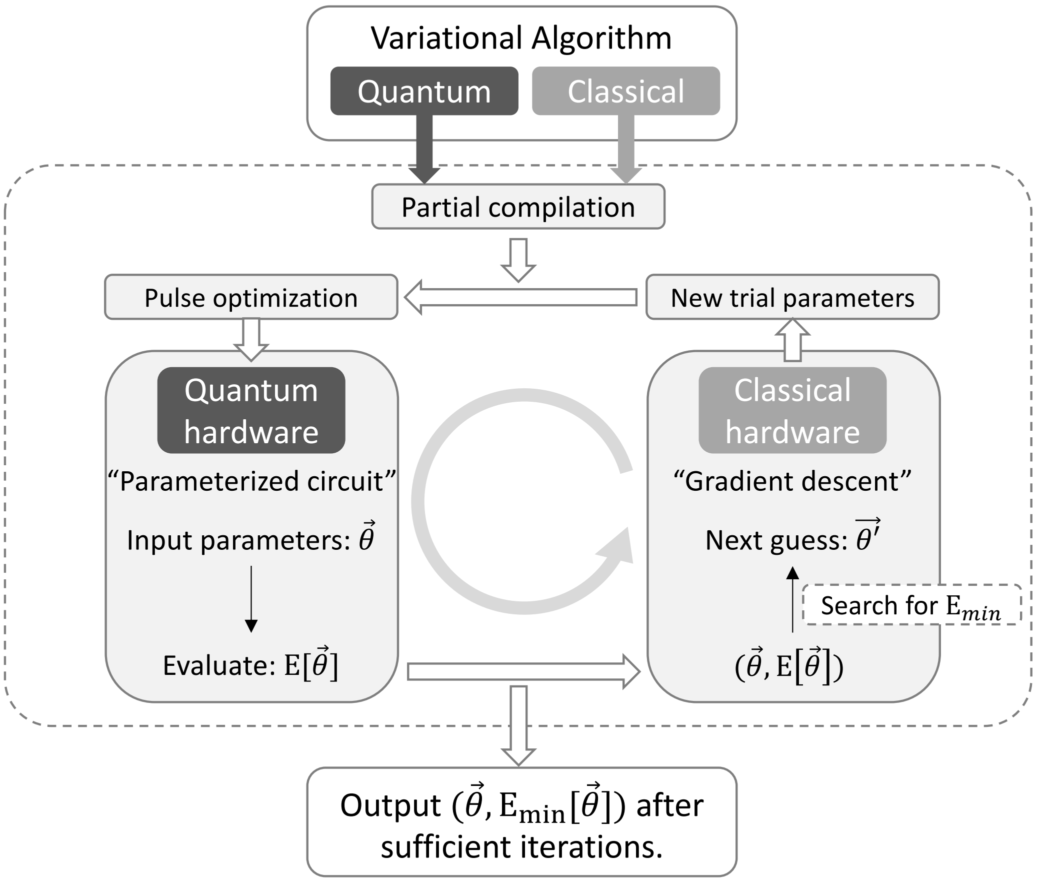

Variational Algorithm
Quantum
Classical
Partial compilation
Pulse optimization
New trial parameters
Quantum hardware
"Parameterized circuit"
Input parameters: $\vec{\theta}$
Evaluate: $E[\vec{\theta}]$
Classical hardware
"Gradient descent"
Next guess: $\vec{\theta'}$
Search for E_{min}
$(\vec{\theta}, E[\vec{\theta}])$
Output $(\vec{\theta}, E_{min}[\vec{\theta}])$ after sufficient iterations.

Figure 3.1: Illustration of a variational quantum algorithm that alternates between a quantum circuit and a classical optimizer. In this process, the quantum circuit (parameterized by $\vec{\theta}$) evaluates some cost function $E[\vec{\theta}]$, and the classical optimizer gradient descends for the next set of parameters.

scaling) in general on a classical computer. This contrast gives rise to a potential quantum speedup for VQE.

The quantum circuit corresponding to the guess is termed an ansatz. While many ansatz choices are possible, Unitary Coupled Cluster Single-Double (UCCSD), an ansatz motivated by principles of quantum chemistry, is considered the gold standard [114, 115]. The UCCSD ansatz is also promising because it could circumvent the barren plateaus issue that affects many other ansatzes [109].

We benchmark the UCCSD ansatz for five molecules: H_2, LiH, BeH_2, NaH, H_2O. These molecules span the state of the art for experimental implementations of VQE: H_2O is the

Molecule	Width (# of Qubits)	# of Params	Gate-Based Runtime
H_2	2	3	35 ns
LiH	4	8	872 ns
BeH_2	6	26	5308 ns
NaH	8	24	5490 ns
H_2O	10	92	33842 ns

Table 3.2: Benchmarked circuits for VQE, using the UCCSD ansatz. Each circuit was optimized, parallel-scheduled, mapped using IBM Qiskit's tools, augmented by an additional optimization pass we wrote to merge consecutive rotation gates. The Gate-Based Runtime is indexed to the pulse durations for each gate reported in Table 3.1.

largest molecule addressed by VQE [91] to date. We generated our UCCSD ansatz circuits using the IBM Qiskit implementation described in [116] as well as the PySCF Python package [117] to manage molecular data.

Both the the circuit depth and number of ansatz parameters in UCCSD scale as $O(N^4)$ in the circuit width [118]. Table 3.2 specifies the exact circuit width, number of variational parameters, and gate-based runtime (circuit depth) for each of the benchmarks. The reported gate-based runtimes are indexed to the pulse durations of each gate reported in Table 3.1. Each circuit was optimized using IBM Qiskit's circuit optimizer pass system, Qiskit's circuit mapper (to conform to nearest neighbor connectivity), and a custom compiler pass to merge neighboring rotation gates on the same axis. We also exploit parallelism to simultaneously schedule as many gates as posisble; the reported gate-based runtimes are for the critical path through the parallelized circuit. These circuit optimizations form a fair baseline for the best circuit runtimes achievable by gate based compilation. Our full circuit optimization code, along with the results of optimization applied to our benchmarks, is available on our Github repository [119].

3.4.2 QAOA

Quantum Approximate Optimization Algorithm (QAOA) is an algorithm for generating approximate solutions to problems that are hard to solve exactly. At an intuitive level, QAOA can be understood as an alternating pattern of Mixing and Cost-Optimization steps. At each Mixing step, QAOA applies diffusion so that every possible state is explored in quantum superposition. At each Cost-Optimization step, a bias is applied to boost the magnitudes of quantum states that minimize a cost function. Thereafter, measuring can yield an approximate solution close to optimal with high probability. The number of alternating Mixing and Cost-Optimization rounds is known as p. Even for small p, QAOA has competitive results against classical approximation algorithms. For example, at $p = 1$, QAOA applied to the NP-hard MAXCUT problem yields a cut of size at least 69% of the optimal cut size [85]. At $p = 5$, simulations have demonstrated that QAOA achieves mean parity with the best-known classical algorithm, Goemans-Williamson, for 10 node graphs [7]. For larger p, QAOA is expected to outperform classical approximation algorithms even for worst-case bounds, although theoretical guarantees have not been established yet. QAOA is of particular interest in the near term because recent work has shown that it is computationally universal [120]. Moreover, QAOA has shown experimental resilience to noise [121]. For these reasons, QAOA is a leading candidate for quantum supremacy [122], the solution of a classically-infeasible problem using a quantum computer.

Similarly to VQE, QAOA is a guess-check-repeat algorithm. In the case of QAOA, the guesses correspond to "Mixing magnitude during iteration $1 \leq i \leq p$" and "Cost-Optimization magnitude during iteration $1 \leq i \leq p$". Hence, the number of parameters in a QAOA circuit is $2p$: one scalar for Mixing magnitude and one for Cost-Optimization magnitude, for each of the p rounds.

We benchmark QAOA for $N = 6$ and 8 node graphs, with the number of QAOA rounds p spanning from 1 to 8. For each (N, p) pair, we benchmark for two types of random graphs:

	$N = 6$		$N = 8$	
	3-Regular	Erdos-Renyi	3-Regular	Erdos-Renyi
$p = 1$	113 ns	84 ns	163 ns	157 ns
$p = 2$	199 ns	151 ns	365 ns	297 ns
$p = 3$	277 ns	223 ns	530 ns	443 ns
$p = 4$	356 ns	296 ns	695 ns	596 ns
$p = 5$	434 ns	368 ns	860 ns	750 ns
$p = 6$	512 ns	440 ns	1025 ns	903 ns
$p = 7$	590 ns	512 ns	1191 ns	1056 ns
$p = 8$	668 ns	584 ns	1356 ns	1209 ns

Table 3.3: Gate-based runtimes for our 32 benchmark QAOA MAXCUT circuits. Our benchmarks consider two types of random graphs: 3-Regular and Erdos-Renyi. We consider both 6 and 8 node graphs–the number of qubits in the circuit is the same as the number of nodes in the graph. We benchmarked over p, the number of repetitions of the basic QAOA block, ranging from 1 to 8, which represents a range of p that is of both theoretical and practical interest [7]. As in Table 3.2, the gate-based runtimes are based on the gate times in Table 3.1, after each circuit has been optimized, parallel-scheduled, and mapped.

3-regular (each node is connected to three neighbors) and Erdos-Renyi (each possible edge is included with 50% probability). This yields $2 \times 8 \times 2 = 32$ benchmarks circuits for QAOA. The gate-based runtimes for each of these benchmarks are reported in Table 3.3. As with the VQE benchmarks, the runtimes are computed after circuit mapping and optimizations, to form a fair baseline.

3.5 GRAPE Compilation

In this section, we describe GRAPE (GRadient Ascent Pulse Engineering), a compilation technique that aims to produce the optimal possible sequence of analog control pulses needed to realize the unitary matrix transformation for a targeted quantum circuit. At an abstract level, GRAPE simply treats the underlying quantum computer as a black box. The black box accepts time-discretized control pulses as input and outputs the unitary matrix of the transformation that is realized by the input control pulses. GRAPE performs gradient descent over the space of possible control pulses to search for the optimal sequence of input signals that achieve the targeted unitary matrix up to a specified fidelity. We used the Tensorflow-based implementation of GRAPE described in [97], which has demonstrated good performance. The gradients are computed analytically and backpropogated with automatic differentiation.

In this chapter, we define the optimal sequence of control pulse as the one of shortest duration–thus, we seek to speed up the pulse time with respect to gate-based compilation. Reducing the pulse time is important in quantum computation because qubits have short lifetimes due to quantum decoherence effects. The decoherence error increases exponential with time, so the effect of a pulse time speedup enters the power of an exponential term. We focus on this error metric because it is one of the dominant error terms for superconducting qubits and it is well understood. However, in principle, GRAPE can be used to control other sources of error such as gate errors, State Preparation and Measurement (SPAM) errors, and qubit crosstalk, as demonstrated in past work [98, 123, 124].

3.5.1 Speedup Sources

Because GRAPE translates directly from a unitary matrix to hardware-level control pulses–without the overhead of an intermediate set of quantum gates–it achieves more optimized control pulses than gate-based compilation does. In particular, we observed significant pulse speedups from GRAPE due to the following factors:

- **ISA alignment**. Gate based compilation incurs a significant overhead because the set of basis gates will not be *directly* implementable on a target machine. For example, while quantum circuits are typically compiled down to CX (CNOT) gates as the default two-qubit instructions, actual quantum computers implement a wide range of native two-qubit operations such as the MS gate or the iSWAP gate. Compiling gates to pulses incurs a significant overhead from this ISA misalignment.

- **Fractional gates**. A unique feature of quantum computing is that all operations can be fractionally performed–for example, $\mathrm{CX}^{1/2}$ is a valid quantum gate, as is CX^p more generally for any power. Often, a fractional application of a basis gate is sufficient to execute a larger quantum operation. The fixed basis set of gate based compilation misses these optimizations, whereas GRAPE works in a continuous basis and realizes fractional gates when beneficial.

- **Control Field Asymmetries**. While gate based compilation puts R_x and R_z gates on an equal footing, at a physical level, there is often a significant asymmetry between the speed and reliability of these operations. As described in A, we model a representative quantum system in which Z-axis qubit rotations are 15 times faster than X-axis qubit rotations. GRAPE's search for the shortest pulse realization will therefore leverage this asymmetry, preferring Z rotations when possible. For example, the H gate is typically implemented by the $R_x(\frac{-\pi}{2})R_z(\frac{-\pi}{2})R_x(\frac{-\pi}{2})$ pulse sequence, which involves two X-axis rotations and one Z-axis rotation. We observe that our GRAPE system instead discovers the equivalent $R_z(\frac{-\pi}{2})R_x(\frac{-\pi}{2})R_z(\frac{-\pi}{2})$ pulse sequence, which only requires one X-axis rotation and therefore executes significantly faster.

- **Maximal circuit optimization**. Although quantum circuits can be optimized at the gate-level by repeatedly applying a set of circuit identity templates, the set of templates must be finite. Opportunities for optimization between distant gates (both in width and depth) may be overlooked. By contrast, GRAPE subsumes all circuit optimizations by working directly in terms of the unitary matrix of the circuit, as opposed to the gate decomposition.

3.5.2 Circuit Blocking for GRAPE

While GRAPE can achieve significant pulse speedups, it is limited by two factors:

- The unitary matrix of the targeted quantum circuit must be specified as input to the GRAPE program. An N-qubit circuit has a $2^N \times 2^N$ matrix (due to the exponential state space of an N-qubit space), which imposes a bound on the maximum circuit size that GRAPE can handle.

- The total convergence time for GRAPE's gradient descent scales exponentially in the size of the target quantum circuit [97]. For example, it typically takes our GRAPE implementation several minutes to find the pulses for a 4 qubit QAOA MAXCUT circuit. Experientially, we also found difficulty consistently finding convergence for deep quantum circuits with $N > 5$ qubits.

For this reason, it is necessary to partition large quantum circuits into blocks of manageable width. We blocked into subcircuits of up to 4 qubits, using the aggregation methodology discussed in [96]. Specifically, we select maximal subcircuits of 4 qubit width, such that partitioning the subcircuit does not delay the execution of subcircuits. This methodology ensures that full GRAPE is strictly better than gate based compilation–otherwise, subcircuits may induce serialization that underperforms gate based compilation. Details are discussed in Section 4.3 of [96].

3.5.3 Binary Search for Minimum Pulse Time

In prior work [97, 96], the pulse length is specified as a static 'upper bound' parameter, `total_time`. Pulse speedups are then performed by adding a term to the cost function that rewards pulses that realize the targeted unitary matrix in time shorter than `total_time`. However, to comply with the automatic differentiation methodology for analytically computing gradients, this cost function term is continuous and rewards *gradual* progress of the pulse towards the target unitary matrix. By contrast, our ultimate goal is to find the *binary* cutoff point specifying the minimal possible time needed to achieve a pulse. Moreover, setting the relative weighting of the speedup term to the fidelity term in the cost function is difficult.

Poor choices of weights can either prevent GRAPE from achieving any speedup or realizing the target fidelity.

As proposed by the prior work [97], our methodology adaptively changes the `total_time` by binary searching for the shortest `total_time` needed to achieve a target unitary matrix. While this incurs the overhead of running on the more iterations [1], it is worthwhile because minimizing the pulse time is exponentially critical in terms of reducing errors.

3.5.4 GRAPE Compilation for QAOA

There are a range of theoretical results setting upper bounds on the circuit complexity needed to achieve a particular quantum operation. For example, it is known that 3 CX gates, sandwiched by single-qubit rotations, is sufficient to implement any two qubit operation. These results were recently generalized to the context of quantum optimal control (a generalization of GRAPE) with a proof that any N-qubit operation can be achieved in $O(4^N)$ time via optimal control [125].

This implies that GRAPE can achieve a significant advantage over gate-based compilation in algorithms like QAOA that have p repeated blocks. While the pulse length from gate-based compilation scales linearly in the p, the GRAPE based pulse length is upper bounded by the maximum time it takes to implement any transformation for an N-qubit circuit. Figure 3.2 demonstrates this behavior for QAOA MAXCUT on the 4-node clique problem. While the pulse length from gate based compilation scales linearly in the number of QAOA rounds p, it asymptotes below 50 ns for GRAPE based pulse lengths. Thus, the pulse speedup advantage of GRAPE increases with p.

As our QAOA benchmarks have circuit widths of 6 and 8 qubits–larger than the 4 qubit blocks we feed to GRAPE–the number of serial blocks will scale linearly with p. Therefore, we don't expect to see an unboundedly growing speedup of GRAPE with increasing p, but

1. Specifically, on the order of $log(M/\Delta t)$ iterations where M is the upper bound on `total_time` and Δt is the desired precision, which we set to 0.3 ns.

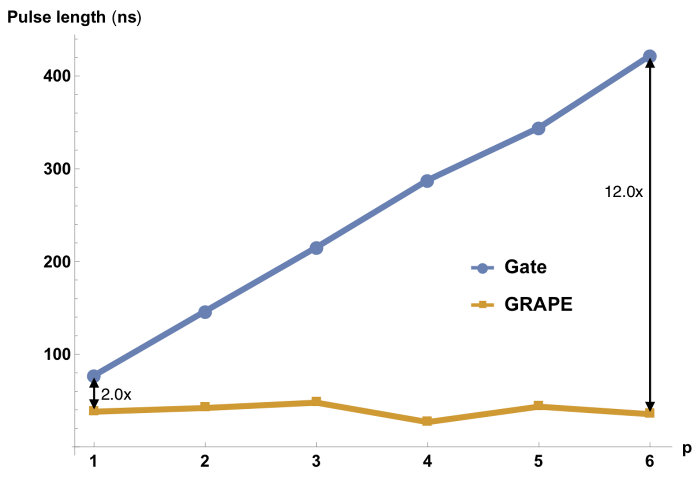


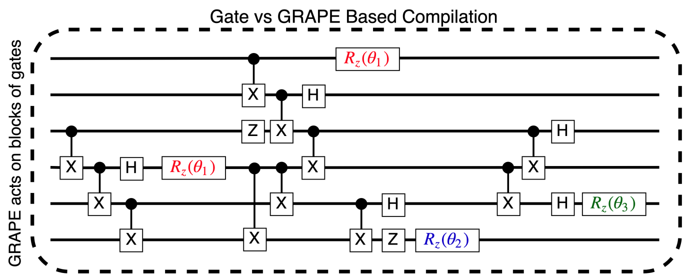
Figure 3.2: Pulse lengths from gate based compilation and full GRAPE for MAXCUT on the 4-node clique. While the gate based pulse times are simply linear in the number of QAOA rounds p, the GRAPE based times asymptote to an upper bound. For each p, a random parametrization was set. The ratio varies from 2.0x at $p = 1$ to 12.0x at $p = 6$.

we still expect to see gains within each 4 qubit block.

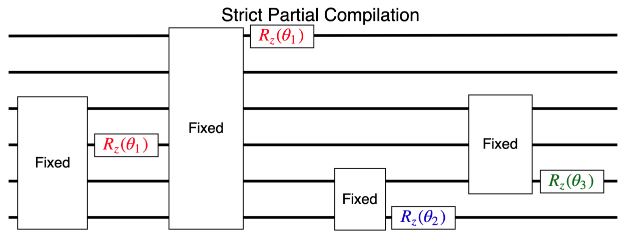
3.6 Strict Partial Compilation

While full quantum optimal control generates the fastest possible pulse sequence for a target circuit, its compilation latency on the order of several minutes is untenable for variational algorithms, in which compilation is interleaved with computation. In order to approach the pulse speedup of GRAPE without incurring the full cost in compilation latency, it is necessary to exploit the structure of the variational circuits. We term this structural analysis as *partial compilation*, and it is executed as pre-computation step prior to executing the variational algorithm on a quantum computer.

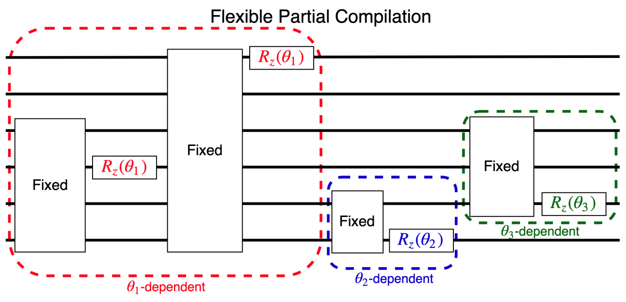
Our first strategy, Strict Partial Compilation, stems from the observation that for typical circuits in variational algorithms, most of the gates are independent of the parametrization. For example, Figure 3.3a shows an example variational circuit. While the circuit has many gates, only four of them depend on the variational θ_i parameters. All of the other gates can be blocked into maximal parametrization-independent subcircuits. Figure 3.3b demonstrates


Gate vs GRAPE Based Compilation
GRAPE acts on blocks of gates
X
Z
H
$R_z(\theta_1)$
$R_z(\theta_2)$
$R_z(\theta_3)$

(a) This is a representative variational circuit, decomposed into gates. In gate-based compilation, each gate is translated by a lookup table to analog control pulses. Compilation amounts to simple concatenation of these control pulses. GRAPE (denoted by the dashed line) considers the unitary matrix for the full circuit and performs gradient descent to find the shortest control pulses that realize the circuit. GRAPE achieves significant pulse speedups, but has substantial compilation latency.



(b) Strict partial compilation blocks the circuit into a strictly alternating sequence of Fixed (parametrization-independent) subcircuits and $R_z(\theta_i)$ gates. Each Fixed subcircuit is precompiled with GRAPE, so that compilation at runtime simply involves concatenating the pulses for each subcircuit.



(c) Flexible partial compilation blocks the circuit into subcircuits that depend on exactly one parameter, θ_i. Parameter monotonicity ensures that these subcircuits have significantly longer depth than the Fixed blocks of strict partial compilation. We use hyperparameter optimization to precompute good hyperparameters (learning rate and decay rate) for each subcircuit. When all θ_i are specified at runtime, we used the tuned hyperparameters to quickly find optimized pulses for each subcircuit.

Figure 3.3: Comparison of compilation strategies. Subfigure (a) depicts gate-based and GRAPE- based compilation for a variational circuit. These two compilation approaches represent opposite ends of a spectrum trading off between between compilation latency and control pulse speedup. We introduce two new compilation strategies, strict and flexible partial compilation, that approach the pulse speedup of GRAPE without the large compilation latency. Subfigures (b) and (c) demonstrates strict and flexible partial compilation respectively.

the application of strict partial compilation to the variational circuit from Figure 3.3a. The sequence of resulting subcircuits is [Fixed, $R_z(\theta_1)$, Fixed, $R_z(\theta_1)$, Fixed, $R_z(\theta_2)$, Fixed, $R_z(\theta_3)$], which exhibits *strict* alternation between 'Fixed' subcircuits that don't depend on any θ_i and $R_z(\theta_i)$ gates that do depend on the parametrization.

After the strict partial compilation blocking is performed, we use full GRAPE to precompute the shortest pulse sequence needed to execute each Fixed subcircuit. These static precompiled pulse sequences can be defined as microinstructions in a low-level assembly such as eQASM [126]. Thereafter, at runtime, the pulse sequence for any parametrization can be generated by simply concatenating the pre-computed pulse sequences for Fixed blocks with the control pulses for each parametrization-dependent $R_z(\theta_i)$ gate. Thus, strict partial compilation retains the extremely fast (essentially instant) compilation time of standard gate based compilation. However, since each Fixed block was compiled by GRAPE, the resulting pulse duration is shorter than if the Fixed blocks had been compiled by gate based compilation. Thus, strict partial compilation achieves pulse speedups over gate-based compilation, with no increase in compilation latency.

Full discussion of the results is deferred to Section 3.8. A priori, we note that the performance of strict partial compilation is tied to the depth of the Fixed subcircuits. For deeper Fixed subcircuits, GRAPE has more opportunities for optimization and can achieve a greater advantage over gate-based compilation. From inspection of Figure 3.3a, we see that the depth of Fixed blocks is determined by the frequency of $R_z(\theta_i)$ gates. For our benchmarked VQE-UCCSD circuits, $R_z(\theta_i)$ gates comprise only 5-8% of the total number of gates, so the Fixed subcircuits have reasonably long depths. For our benchmarked QAOA circuits however, the $R_z(\theta_i)$ gates comprise 15-28% of the total number of gates, so the Fixed subcircuits have short depths and the potential advantage of strict partial compilation is limited. This motivates us to consider other strategies that more closely match the pulse speedups of full GRAPE.

3.7 Flexible Partial Compilation

As strict partial compilation is bottlenecked by the depth of Fixed subcircuits, we are motivated to consider strategies that create deeper subcircuits. The core idea behind flexible partial compilation is to create subcircuits that are only 'slightly' parametrized, in that they depend on at most one of the θ_i variational parameters. As discussed below, we can perform hyperparameter tuning to ensure that GRAPE finds optimized pulses for single-angle parametrized subcircuits much faster than for general subcircuits.

3.7.1 Parameter Monotonicity

An initial strategy for creating these single-angle parametrized subcircuits would be to merge each consecutive pair of Fixed and $R_z(\theta_i)$ subcircuits into a single subcircuit that only depends on θ_i. However, this strategy would add at most one gate of depth to each subcircuit, which would not lead to significantly better pulses. However, we make a key observation which we term *parameter monotonicity.* For both the VQE UCCSD and QAOA circuits, the appearances of θ_i-dependent gates is monotonic in i–once a θ_i dependent gate appears, the subsequent parametrization-dependent gates must be θ_j for $j \geq i$. As a result, subcircuits with the same value of θ_i must be *consecutive.* For example, the sequence of angles in parametrization-dependent gates could be $[\theta_1, \theta_1, \theta_2, \theta_3]$ as in Figure 3.3a, but not $[\theta_1, \theta_2, \theta_3, \theta_1]$.

At a high level, parameter monotonicity for VQE/UCCSD and QAOA arise because their circuit constructions sequentially apply a circuit corresponding to each parameter exactly once. For instance, in QAOA, each parameter corresponds to the magnitude of Mixing or Cost-Optimization during the ith round–once the corresponding Mixing or Cost-Optimization has been applied, the circuit no longer depends on that parameter. Parameter monotonicity is not immediately obvious from visual inspection of variational circuits, because the circuit constructions and optimizations transform individual θ_i-dependent gates to ones that are

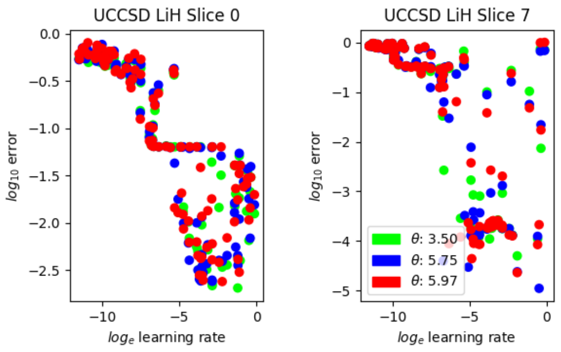


Figure 3.4: The 0th single-angle dependent subcircuit of the UCCSD LiH circuit has two angle dependent gates, the 7th has eight. These four qubit circuits are representative of the circuits studied in this work as well as larger future circuits due to the necessity of circuit blocking for circuits with more than four qubits. The graphs above plot GRAPE error against ADAM learning rate. For each permutation of the argument of the angle dependent gates in the subcircuits, the same range of learning rate values achieves the lowest error.

parametrized in terms $-\theta_i$ or $\theta_i/2$. We resolve these latent dependencies by explicitly tagging the dependent parameter in software during the variational circuit construction phase.

The implication of parameter monotonicity is that the subcircuits considered by flexible partial compilation are significantly deeper than the ones considered by strict partial compilation. Figure 3.3c demonstrates a small example; note that the θ_1-dependent subcircuit indicated by red dashed lines is significantly deeper than the subcircuits generated by strict partial compilation.

3.7.2 Hyperparameter Optimization

In GRAPE, an optimal control pulse is one that minimizes a set of cost functions corresponding to control amplitude, target state infidelity, and evolution time, among others[97]. To obtain an optimal control pulse, the GRAPE algorithm manipulates a set of time-discrete control fields that act on a quantum system. It may analytically compute gradients of the cost functions to be minimized with respect to the control fields. These gradients are used to update control fields with an optimizer such as ADAM or L-BFGS-B. As opposed to the control fields, which are parameters manipulated by GRAPE, these optimizers have their own parameters such as learning rate and learning rate decay. These parameters are termed hyperparameters because they are set before the learning process begins.

Because they are inputs to the learning process, there is no closed form expression relating hyperparameters and the cost functions a learning model is minimizing. This makes hyperparameter optimization an ideal candidate for derivative free optimization techniques. Recent work has shown that tuning hyperparameters with methods such as bayesian optimization and radial basis functions can significantly improve performance for stochastic and expensive objectives such as minimizing the training error of neural networks [127, 128]. In our work, we employ hyperparameter optimization on GRAPE's ADAM optimizer. We realize faster convergence to a desired error rate over the baseline, significantly reducing compilation latency.

In particular, we make the observation that a high-performing hyperparameter configuration for a single-angle parameterized subcircuit is robust to changes in the argument of its θ_i-dependent gates, as shown in Figure 3.4. Therefore, we are able to precompute high-performing hyperparameter configurations for each single-angle parameterized subcircuit and employ them in compilation. For each iteration of a variational algorithm, the argument of the θ_i-dependent gates of each subcircuit will change, but the same hyperparameters are specified to GRAPE's optimizer, maintaining the same reduced compilation latency.

3.8 Results

Our results were collected using over 200,000 CPU-core hours on Intel Xeon E5-2680 processors, using up to 64 GB of memory per GRAPE process. The large volume of compute is a result of both the high cost of running GRAPE and the number and large circuit size of benchmarks. We fixed randomization seeds when appropriate for both reproducability and consistency between identical benchmarks. Our results are available in Jupyter notebooks on our Github

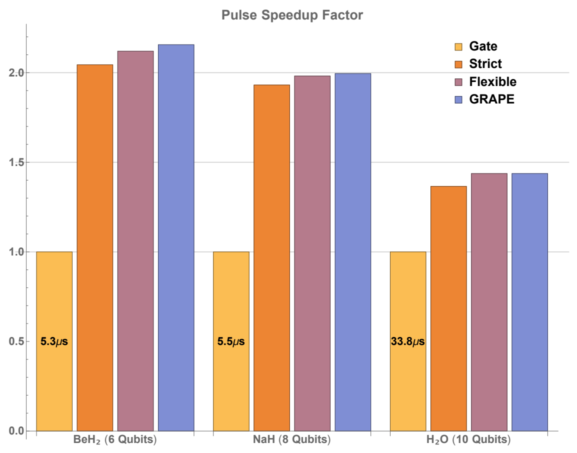


Figure 3.5: Pulse speedup factors (relative to gate based compilation) for VQE circuits. Full QOC 1.5-2x reductions in pulse durations for these circuits. Strict and flexible partial compilation recover 95% and 99% of this speedup respectively. Detailed results are reported in Table 3.4.

repository [119].

3.8.1 Pulse Speedups

Compilation	UCCSD					Max-Cut							
						3-Regular, N=6		Erdos-Renyi, N=6		3-Regular, N=8		Erdos-Renyi, N=8	
Techniques	H_2	LiH	BeH_2	NaH	H_2O	p=1	p=5	p=1	p=5	p=1	p=5	p=1	p=5
Gate-based	35.3	871.1	5308.3	5490.4	33842.2	113.2	433.6	83.7	367.8	162.5	860.0	157.1	749.5
Strict Partial	15.0	307.0	2596.5	2842.7	24781.4	91.2	397.6	54.0	291.8	134.0	711.6	100.0	551.7
Flexible Partial	5.0	84.0	2503.8	2770.8	23546.7	72.0	206.2	26.4	150.0	112.0	498.9	80.5	434.8
Full GRAPE	3.1	19.3	2461.7	2752.0	23546.7	72.0	179.0	26.6	141.2	112.0	498.9	81.6	513.7

Table 3.4: Experimental results for pulse durations (in nanoseconds) across the VQE-UCCSD and QAOA benchmarks.

Figure 3.5 shows the pulse times speedup factors across our QAOA benchmarks for partial compilation and for full GRAPE, normalized to the gate-based compilation baseline. We present the normalized speedup factors, because the H_2O VQE-UCCSD benchmark is 10x larger; the raw pulse times are presented in Table 3.4.

For the BeH_2 and NaH VQE-UCCSD benchmarks, full GRAPE gives a 2.15x and 2.00x speedup in pulse duration respectively. Strict partial compilation is able to recover almost

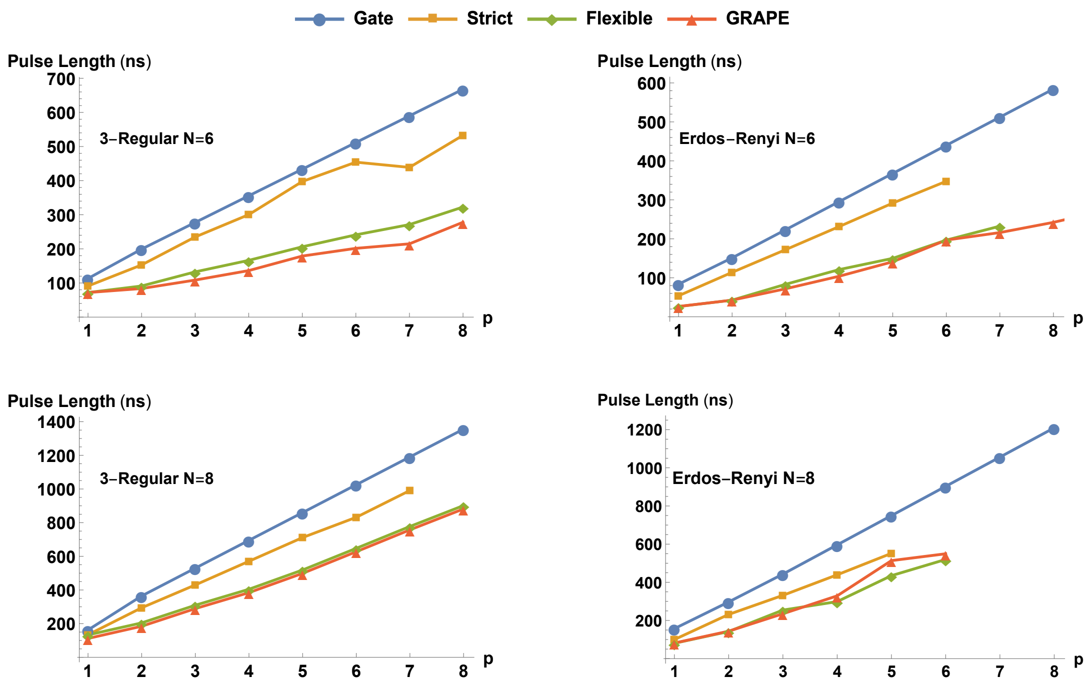


Figure 3.6: Pulse durations for QAOA MAXCUT benchmarks under the four compilation techniques, across all benchmarks. The gate based pulse time always increases linearly in p, the number of repeated rounds in the QAOA circuit. The average GRAPE pulse speedup is 2.6x for 6-node graphs and 1.8x for 8-node graphs. Strict partial compilation only achieves a modest speedup over gate based compilation, but flexible partial compilation essentially matches the GRAPE speedup exactly. The omitted data points correspond to computations that did not complete in 12 CPU-core hours, even after parallelizing subcircuit jobs.

this full advantage, with speedups at 2.04x and 1.93x respectively. As discussed in Section 3.6, this matches the expectations, because the VQE-UCCSD benchmarks have relatively deep Fixed subcircuits. Finally, the speedups for flexible partial compilation are 2.12x and 1.98x, which nearly closes the gap between strict partial compilation and GRAPE.

The H_2O benchmark has similar relative speedups between strict, flexible, and GRAPE, with factors of 1.37, 1.44, 1.44.[2] However, the advantage over gate based compilation is smaller than for the BeH_2 and NaH benchmarks.

Figure 3.6 shows results for QAOA benchmarks. Strict partial compilation has speedups

2. In fact, the pulse speedup for flexible partial compilation exactly matches GRAPE, because each 4-qubit block handled by GRAPE depends on at most one parameter.

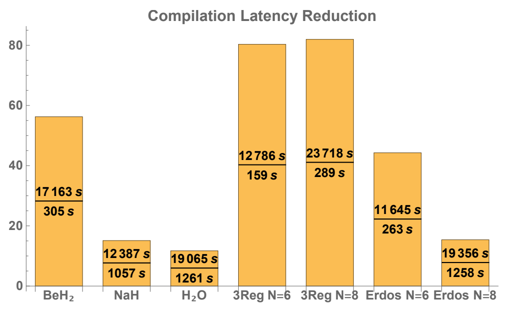


Figure 3.7: Reduction factors in compilation latency. The ratios indicate the average compilation latency using flexible partial compilation divided by latency using full GRAPE compilation. Flexible partial compilation uses about an hour of pre-compute time to determine the best learning rate - decay rate pair for each subcircuit.

of 1.22x and 1.33x across the $N = 6$ and $N = 8$ qubit benchmarks respectively. By contrast, flexible partial compilation has average speedups of 2.3x and 1.8x across the $N = 6$ and $N = 8$ benchmarks, which almost matches the results from GRAPE. This separation between strict and flexible partial compilation matches the expected results discussed in Section 3.6. The high frequency of parametrized gates in QAOA limits the depth of Fixed blocks, so strict blocking has limited mileage. However, due to the four-qubit maximum subcircuit size for GRAPE, each block will rarely depend on more than one θ_i parameter. On these single-angle dependent blocks, flexible partial compilation achieves the same pulse speedups as GRAPE.

3.8.2 Compilation Latency Reduction

Figure 3.7 shows the compilation latency reduction achieved by flexible partial compilation, relative to full GRAPE compilation. As described in Section 7.2, flexible partial compilation is able to dramatically speed up the gradient descent's convergence by tuning the learning rate and decay rate hyperparameters on a per-subcircuit basis. We note that the 3-regular graphs achieve particularly high compilation latency reduction factors of 80.3x and 81.9x. Across all benchmarks, the reduction in compile time is from hours to minutes, which is critical in the context of variational algorithms.

3.8.3 Simulation with Realistic Pulses

While we performed our GRAPE runs without accounting for error or noise sources for simplicity, it can be adapted to account for these sources. For example, we could demand well-shaped pulses, account for leakage into higher states outside the binary qubit abstraction, or explicitly model the qubit decoherence times. To demonstrate that these sources can be accounted for, we re-ran two of our VQE and QAOA benchmarks with Full GRAPE using these more realistic assumptions:

- Allowing only 1 pulse datapoint every nanosecond (1 GSa/s), versus 20 GSa/s in the other results presented in this chapter.
- Including leakage into the qutrit *leakage* level. Other results in this chapter use the binary-qubit approximation, as outlined in the system Hamiltonian in Section 3.11.
- Application of aggressive pulse regularization in GRAPE to ensure that the pulse shapes follow a Gaussian envelope and have smooth 1st and 2nd derivatives.

Table 3.4 compares the pulse speedups due to GRAPE, under both our standard (less realistic) GRAPE settings and under the more realistic settings that account for the three items above. For VQE and QAOA applications, the GRAPE speedups speedups are 11.4x (standard) vs 8.8x (realistic) and 4.5x (standard) vs. 3.0x (realistic) respectively. While the more realistic pulses do seem to have somewhat lower pulse speedup factors, they are similar and still feature significant speedup over gate based compilation.

3.8.4 Aggregate Impact on Total Runtime

For a quantitative sense of aggregate impact, we note that VQE requires thousands of iterations, even for small molecules. For instance, past work in VQE, towards estimating the ground state energy of BeH_2, required 3500 iterations [90]. Per Figure 3.7, this would amount

Gate ns $\rightarrow$ GRAPE ns (reduction)	H_2 VQE	Erdos-Renyi $N = 3$
Standard	$35.3 \rightarrow 3.1$ (11.4x)	$15.0 \rightarrow 3.3$ (4.5x)
More Realistic	$420 \rightarrow 48$ (8.8x)	$285 \rightarrow 96$ (3.0x)

Table 3.5: Speedups due to GRAPE compilation under the standard settings and under more realistic settings, which account for lower sampling rates, qutrit leakage, and pulse regularization. Results are given for the H_2 VQE benchmark and for the Erdos-Renyi $N = 3$ QAOA benchmark. The speedup factors due to GRAPE are similar with and without the more realistic assumptions.

to over 2 years of runtime compilation latency via Full-GRAPE. By contrast, strict partial compilation achieves zero runtime compilation latency via lookup table, and the pre-computed pulses for the fixed blocks were compiled in under 1 hour. Since the UCCSD ansatz has quartic scaling in the number of parameters [114], the number of iterations required scales aggressively for bigger molecules and the advantage of our approach will scale favorably. Further experimental work is needed to estimate the advantage of our approach for larger molecules in terms of total runtime, but extrapolation from small molecules BeH_2 seems promising. Similarly, while the improvement in quality-of-result due to the shorter pulse times from GRAPE is difficult to quantify without concrete experiments, we emphasize that the error due to decoherence scales exponentially with quantum runtime. Therefore, we again expect favorable results, owing to the significant pulse time speedup of our techniques relative to gate based compilation.

3.9 Conclusion

Variational quantum algorithms such as VQE and QAOA are strong candidates for demonstrating a quantum advantage in problems such as molecular ground state estimation, MAXCUT approximation, and prime factorization. Unlike prior algorithms, variational algorithms are parametrized, with the parameters at each iteration determined based on the results of previous iterations. Consequently, compilation is interleaved with computation. As a result, it is not practical to each variational circuit with out-of-the-box GRAPE, which takes several minutes to find an optimized pulse even on small (4-qubit) circuit.

Our partial compilation techniques offer a path to achieving the pulse speedups of GRAPE, without incurring its compilation latency. On the VQE-UCCSD circuits, our strict partial compilation strategy achieves 1.5x-2x pulse speedups over gate based compilation, almost matching the speedups from full GRAPE. Strict partial compilation is performed by precomputing optimal pulses for Fixed blocks. During execution, it has the same–essentially instant–lookup table based compilation procedure as gate based compilation. Thus, strict partial compilation is strictly better than gate based compilation.

For QAOA circuits, while strict partial compilation only achieves modest pulse speedup, we find that flexible partial compilation almost exactly matches the pulse speedups of GRAPE. Flexible partial compilation precomputes the best hyperparameters for each slice, so that when the θ_i parameters are specified at runtime, an optimized pulse sequence can be computed rapidly. For our benchmarked circuits, we found 10-100x reductions in compilation latency from flexible partial compilation, relative to full GRAPE compilation.

We emphasize that achieving optimized pulses is critical because error due to decoherence error is exponential in the pulse duration. Thus, our pulse speedups are not merely about wall time speedups for quantum circuits, but moreso about making computations possible in the first place, before the qubits decohere.

3.10 Future Work

The industry adoption of the OpenPulse standard will usher an experimental era for pulse-level control. Running our partial compilation schemes on an actual machine will be valuable in terms of determining exactly how to weigh tradeoffs between pre-computation resources, compilation latency, and pulse durations.

On the computational side, we also see significant potential for extending the scalability of GRAPE. While past work has successfully used GRAPE on 10 qubit widths with very simple circuits (for example, 10 identical single-qubit rotations in parallel), we found that for complicated circuits, GRAPE only converges reliabily with widths up to 4 qubits. This 4-qubit blocking width limits the depths of the subcircuits that both GRAPE and our partial compilation schemes can consider. For example, in the additional two VQE-UCCSD molecules benchmarks (H_2 and LiH) reported in Table 4, flexible partial compilation and full GRAPE achieve 7-50x pulse speedups because the benchmarks are 2 and 4 qubits in width. Thus, investigating the convergence properties of GRAPE and extending the circuit widths it reliably converges for will substantially extend the advantage that these techniques can achieve over gate based compilation.

3.11 System Hamiltonian

Although our techniques are general and apply to any quantum computer, the pulses produced by GRAPE are specific to the underlying hardware platform. We chose to compile to control pulses for a quantum computer with gmon superconducting qubits [99], because this qubit type is one of the leader contenders for scalable quantum machines. For instance, the gmon qubit is central to Google's experimental efforts for demonstrating quantum supremacy.

The control pulse inputs that we specified to GRAPE were based on the gmon's system Hamiltonian. Each qubit, j, has a flux-drive control pulse and a charge-drive control pulse which have respective Hamiltonians, truncated to the qubit subspace:

$$H_{c,j}(t) = \sum_{j=1}^{N} \Omega_{c,j}(t)(a_j^\dagger + a_j) = \sum_{j=1}^{N} \Omega_{c,j}(t) \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

and

$$H_{f,j}(t) = \sum_{j=1}^{N} \Omega_{f,j}(t)(a_j^\dagger a_j) = \sum_{j=1}^{N} \Omega_{f,j} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

It can be seen from exponentiating these matrices that the control pulses correspond to $R_x(\theta)$ and $R_z(\phi)$ type gates respectively. We chose maximium drive amplitudes of $|\Omega_{c,j}(t)| \leq 2\pi \times 0.1$ GHz and $|\Omega_{f,j}(t)| \leq 2\pi \times 1.5$ GHz. These values, including the asymmetry between charge and flux drive, are representative of typical machines.

In addition to these single qubit terms, there is a control pulse for each pair of connected qubits. We consider a rectangular-grid topology with nearest-neighbor connectivity. Between each connected pair of qubits j and k, the corresponding control Hamiltonian is

$$H_{j,k}(t) = g(t)(a_j^\dagger + a_j)(a_k^\dagger + a_k)$$

This two-qubit interaction type corresponds to the entangling iSWAP gate (which swaps two qubits and also applies a phase factor). We use a maximum coupling strength of $|g(t)| \leq 2\pi \times 50$ MHz

Within the GRAPE software, we discretized the control pulses to 0.05 ns time slices. We set a target fidelity of 99.9% for each invocation of GRAPE. Raw data from all of our GRAPE runs are available at our Github repository [119].

CHAPTER 4
MINIMIZING STATE PREPARATIONS IN VARIATIONAL QUANTUM EIGENSOLVER BY PARTITIONING INTO COMMUTING FAMILIES

Variational quantum eigensolver (VQE) is a promising algorithm suitable for near-term quantum machines. VQE aims to approximate the lowest eigenvalue of an exponentially sized matrix in polynomial time. It minimizes quantum resource requirements both by co-processing with a classical processor and by structuring computation into many subproblems. Each quantum subproblem involves a separate state preparation terminated by the measurement of one Pauli string. However, the number of such Pauli strings scales as N^4 for typical problems of interest—a daunting growth rate that poses a serious limitation for emerging applications such as quantum computational chemistry. We introduce a systematic technique for minimizing requisite state preparations by exploiting the simultaneous measurability of partitions of commuting Pauli strings. Our work encompasses algorithms for efficiently approximating a MIN-COMMUTING-PARTITION, as well as a synthesis tool for compiling simultaneous measurement circuits. For representative problems, we achieve 8-30x reductions in state preparations, with minimal overhead in measurement circuit cost. We demonstrate experimental validation of our techniques by estimating the ground state energy of deuteron on an IBM Q 20-qubit machine. We also investigate the underlying statistics of simultaneous measurement and devise an adaptive strategy for mitigating harmful covariance terms.

4.1 Introduction

The present Noisy Intermediate-Scale Quantum (NISQ) era [16] is distinguished by the advent of quantum computers comprising tens of qubits, with hundreds of qubits expected in the next five years. Although several thousand logical error-corrected qubits, backed by millions of device-level physical qubits, are needed to realize the originally-envisioned quantum applications such as factoring [10] and database search [9], a new generation of *variational* algorithms have been recently introduced to match the constraints of NISQ hardware.

Variational Quantum Eigensolver (VQE) [84] is one such algorithm that is widely considered a top contender, if not the top contender, for demonstrating a useful quantum speedup. VQE is used to approximate the lowest eigenvalue of a matrix that is exponentially sized in the number of qubits. This is a very generic eigenvalue problem with a wide class of applications such as molecular ground state estimation [84]; maximum 3-satisfiability, market split, traveling salesperson [129]; and maximum cut [107]. In this chapter, we focus on the molecular ground state estimation problem which has already been demonstrated experimentally, though we underscore that the full range of VQE applications is very broad.

VQE solves a similar problem as Quantum Phase Estimation (QPE) [130, 131], an older algorithm that requires large gate counts and long qubit coherence times that are untenable for near-term quantum computers. VQE mitigates these quantum resource requirements by shifting some computational burden to a classical co-processor. As a result, VQE achieves low gate count circuits and error resilience, but at the cost of requiring many iterations where each iteration measures one of $O(N^4)$ terms.

This is a daunting scaling factor that poses practical limitations. It was observed that this N^4 scaling could be partly mitigated by performing simultaneous measurement: when two terms correspond to commuting observables, they can be measured in a single state preparation. Our work starts from this observation and we seek to exploit this idea to minimize the total number of state preparations needed.

Our specific contributions include:

1. Efficient approximation algorithms for partitioning the N^4 terms into commuting families, i.e. approximating the MIN-COMMUTING-PARTITION.

2. A circuit synthesis tool for simultaneous measurement.

3. Statistical analysis of simultaneous measurement and a procedure for guarding against harmful covariance terms.

4. Validation of these techniques through benchmarks, simulations, and experiments.

The rest of this chapter is structured as follows. Section 4.2 presents relevant background material and Section 4.3 surveys prior work. Section 4.4 analyzes the commutativity of the terms of interest (Pauli strings) and Section 4.5 presents a technique for minimizing the number of state preparations by mapping MIN-COMMUTING-PARTITION to a MIN-CLIQUE-COVER instance that can be approximated. Section 4.6 develops an alternate technique that takes advantage of molecular Hamiltonian structure in order to approximate the MIN-COMMUTING-PARTITION with minimal classical overhead.

Section 4.7 shows and analyzes the circuit synthesis procedure that allows simultaneous measurements between commuting Pauli strings. Section 4.8 presents results for our techniques on benchmark molecules and Section 4.9 demonstrates experimental validation. Section 4.10 studies the underlying statistics and discusses a strategy for detecting and correcting course if a partition is harmed by covariance terms. We make concluding remarks and propose future work in Section 4.11.

4.2 Background

We assume an introductory-level knowledge of quantum computing. We refer newer readers to one of many excellent resources such as [132], [133], or [32].

4.2.1 Quantum Measurement

A standard procedure in quantum algorithms is to measure a qubit. In hardware, the standard measurement that can be performed is a measurement in the Z-basis, or computational basis. Figure 4.1 depicts such a measurement. The qubit's state is a point on the surface of the

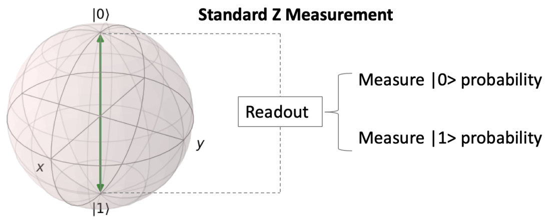


Figure 4.1: Z-basis (computational basis) measurement of a qubit yields $|0\rangle$ or $|1\rangle$ with a probability corresponding to the latitude of the qubit on the Bloch sphere.

Bloch sphere—states with northern latitudes are close to the $|0\rangle$ state and southern latitudes are close to the $|1\rangle$ state. Measurement, or readout, causes the qubit to collapse to either the $|0\rangle$ or $|1\rangle$ state, with a probability dependent on the latitude.

At a more mathematical level, the deeper meaning of measuring a qubit in the

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

basis is to project the qubit's state onto the eigenvectors of the Z operator, which are $|0\rangle$ and $|1\rangle$. In the same sense, we can measure other observables, such as the other two *Pauli matrices*:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \text{and} \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

The eigenvectors of X are termed $|+\rangle$ and $|-\rangle$, and they are antipodal points along X-axis of the Bloch sphere. Similarly, Y's eigenvectors, $|i\rangle$ and $|-i\rangle$, are antipodal along the Y-axis. Since hardware cannot directly measure along these axes, measurements of X (Y) are performed by first rotating the Bloch sphere with a unitary matrix so that the X (Y) -axis becomes aligned with the Z-axis. These rotations are depicted in Figure 4.2. Subsequently, a standard Z-basis measurement can be performed, whose outcome can then be mapped to an

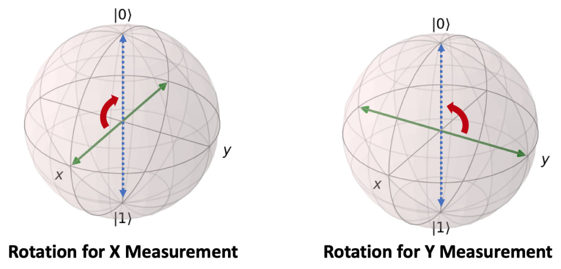


Figure 4.2: Measurement of the X or Y Pauli matrices requires us to first apply a unitary rotation operation that rotates the X or Y axis to align with the Z axis. Subsequently, a standard Z-basis measurement yields the outcome of the X or Y measurement.

effective X (Y) measurement.

The specific rotation that accomplishes the X-to-Z axis change is the $R_y(-\pi/2)$ transformation, which is typically captured in quantum circuits by the similar H gate/matrix. The Y-to-Z axis change is accomplished by the $R_x(\pi/2)$ transformation, which is typically captured [134] in quantum circuits by the $HS^\dagger$ gates/matrix.

The same general principle applies towards measuring observables across multiple qubits: measurement is accomplished by applying a quantum circuit that rotates the eigenvectors of the target observable onto the computational basis vectors. The unitary matrix for such a transformation is simply the one that has the orthonormal eigenvectors of the observable as column vectors. In our study, we will be interested in measuring Pauli strings, which are tensor products of Pauli matrices across multiple qubits.

4.2.2 Simultaneous Measurement and Commutativity

From the preceding discussion, we can see that two observables can be measured simultaneously if they share a common eigenbasis, i.e. they are simultaneously diagonalizable. In this case, they can be measured simultaneously by applying the unitary transformation that rotates their shared eigenbasis onto the computational basis. In the case of Hermitian operators, such as the Pauli strings of interest to us, two observables share an eigenbasis if and only if they commute [135, Chapter 1], i.e. the order of their product is interchangeable.

Moreover, this relationship extends beyond simple pairs: given a family of pairwise commuting observables, there exists a shared eigenbasis that simultaneously diagonalizes *all* of the observables (rather than it merely being a situation in which *each* pair has a separate shared eigenbasis) [136, Theorem 1.3.21].

In this chapter, we will exploit this property to simultaneously measure multiple Pauli string observables with a single state preparation and measurement circuit. Notice that this problem is non-trivial because commutativity is not transitive (and hence, *not* an equivalence class). Consequently, finding optimal partitions of commuting families is a hard problem, as we formalize later.

4.2.3 Quantum Computational Chemistry

Quantum computational chemistry has been a long targeted problem on the classical computer. Due to the limits of classical computing resources, we are only able to perform *approximate* classical simulations. Examples include Hartree Fock ($O(N^4)$ runtime [137], only takes ground state orbitals into account), Density Functional Theory ($O(N^3)$ runtime [138], but with even less precision), and Coupled Cluster Single-Double ($O(N^6)+$ runtime [139], only considers single and double excitations).

The way to achieve chemical accuracy is to use Full CI (full configuration interactions), which considers all necessary orbitals. Classically this will generally require $O(\binom{M}{N}) \rightarrow$ exponential runtime [140]. On the other hand, quantum computation is able to encode an exponential amount of molecular information into a polynomial number of qubits and thereby achieve Full CI in polynomial time [141].

4.2.4 Variational Quantum Eigensolver (VQE)

As mentioned previously, VQE can be applied to a wide class of problems that are solvable as minimum-eigenvalue estimation [129, 107]. In this chapter, we focus on the application that has received the most commercial and experimental interest: estimating molecular ground state energy. Within the molecular context, we use VQE to approximate the lowest eigenvalue of a matrix called the Hamiltonian that captures the molecule's energy configuration. The lowest eigenvalue is the ground state energy which has important implications in chemistry such as determining reaction rates [110] and molecular geometry [111].

The Hamiltonian matrix for a molecule can be written in the second quantized fermionic form as [109]

$$H = \sum_{p=1}^{N}\sum_{q=1}^{N} h_{pq} a_p^\dagger a_q + \sum_{p=1}^{N}\sum_{q=1}^{N}\sum_{r=1}^{N}\sum_{s=1}^{N} h_{pqrs} a_p^\dagger a_q^\dagger a_r a_s \tag{4.1}$$

where $a^\dagger$ (a) is the fermionic raising (lowering) operator, and N is the number of qubits and also the number of molecular basis wavefunctions considered. The h_{pq} and h_{pqrs} terms can be computed classically via electron integral formulas implemented by several software packages [142, 117, 143]. The second sum in Equation 4.1 indicates that the fermionic form of the Hamiltonian has $O(N^4)$ terms [144, 145]. It can be translated to qubit form by an encoding such as Jordan-Wigner [146], Parity [147], or Bravyi-Kitaev [148], as we will discuss further in Section 4.6. The resulting qubit form will also have $O(N^4)$ terms, where each term is a Pauli string.

It is difficult to directly AND efficiently estimate $\langle H\rangle$, the expected energy of the Hamiltonian under an input state vector. The approach of VQE is to estimate it *indirectly* but efficiently, by employing linearity of expectation to decompose $\langle H\rangle$ into a sum of $O(N^4)$ expectations of Pauli strings, which can each be computed efficiently. In the standard and original formulation of VQE, each of these Pauli strings is measured via a separate state preparation [84].

At its core, VQE can be described as a guess-check-repeat algorithm. Initially, the algorithm *guesses* the minimum energy eigenvector of the Hamiltonian H. Then, it *checks* the actual energy for the guessed eigenvector by summing expected values over the $O(N^4)$

directly measurable Pauli strings, as previously described. Finally, it *repeats* by trying a new guess for the minimum energy eigenvector, with the assistance of a classical optimizer that guides the next guess based on past results. The potential quantum speedup in VQE arises from the fact that checking the energy on a classical computer would require matrix multiplication of an exponentially-sized state vector; by contrast, the energy can be estimated efficiently with a quantum computer by summing over the expected values of the $O(N^4)$ Pauli strings.

Algorithm 2: Variational Quantum Eigensolver (VQE)

```
Result: Approximate ground state energy, $\min_{\vec{\theta}} \langle H \rangle_{\psi(\vec{\theta})}$
$\vec{\theta_1} \leftarrow$ random angles
$i \leftarrow 1$
while (not classical optimizer termination condition) do
    for $j \in [O(N^4)]$ do
        for $O(1/\epsilon^2)$ repetitions do
            Prepare $\psi(\vec{\theta_i})$
            Measure $\langle H_j \rangle_{\psi(\vec{\theta_i})}$
        end
    end
    $\langle H \rangle_{\psi(\vec{\theta_i})} \leftarrow \sum_j \langle H_j \rangle_{\psi(\vec{\theta_i})}$
    Record $(\theta_i, \langle H \rangle_{\psi(\vec{\theta_i})})$
    $i$++
    Pick new $\theta_i$ via classical optimizer
end
```

Algorithm 2 presents the pseudocode for VQE, under the standard 'Naive' formulation where each Pauli string is measured separately. The resource complexity of VQE is clear from this code: the inner `for` loops run $O(N^4/\epsilon^2)$ times and each iteration requires a separate state preparation and measurement. The outer `while` loop termination condition is dependent on both the classical optimizer and the ansatz–we discuss the latter next.

4.2.5 Unitary Coupled Cluster Single Double Ansatz

Since the number of possible state vectors spans an exponentially large and continuous Hilbert space, we seek to restrict the family of candidate energy-minimizing states. Such a family is called an ansatz, and the ansatz state $|\psi(\vec{\theta})\rangle$ is parametrized by a vector of independent parameters, $\vec{\theta}$. Since VQE aims to run in polynomial time, the number of parameters should be polynomial. While our work in this chapter is applicable to any ansatz, we focus our attention to the Unitary Coupled Cluster Single Double (UCCSD) ansatz, which has generally been the leading contender for molecular ground state estimation. In addition to having a sound theoretical backing (the coupled cluster approach is the gold standard for computational chemistry [109, 115]), UCCSD is more resilient to barren plateaus in the optimization landscape that are experienced by hardware-oriented ansatzes [149, 109]. Recent work has also demonstrated the experimental superiority of UCCSD to other ansatz types [150].

In terms of the number of qubits (which is also the number of molecular basis wavefunctions) N, the total gate count of UCCSD is $O(N^4)$ [151, 152], which can be parallelized in execution to $O(N^3)$ circuit depth. As a concrete scaling example, a recent 4-qubit, 2-electron UCCSD circuit construction required circuit depth of 100 gates, spanning 150 total gates [150]. This is already out of range of present machines—the experimental work thus far has required many symmetry reductions and approximations to implement UCCSD. The number of parameters in UCCSD, with respect to the number of electrons and wavefunctions is $O(N^2\eta^2)$, or $O(N^4)$ under the standard assumption that these two terms are asymptotically related by a constant.

4.2.6 Mutually Unbiased Bases

Finally, we give a brief overview of Mutually Unbiased Bases (MUB) [153, 154], a concept in quantum information theory that is connected to our overarching question of maximizing the information learned from a single measurement. In the case of qubits, MUBs describe a partitioning of the $4^N - 1$ N-qubit Pauli strings (Identity is excluded) into commuting families of maximal size. For example, Table 4.1 shows a MUB for the 2-qubit Pauli strings. Notice that each row corresponds to a commuting family. Also note that not all rows are created equal–in the first three rows, the shared eigenbasis features separable eigenvectors. In the last two rows, the shared eigenbasis has entanglement between the two qubits.

Operator 1	Operator 2	Operator 3	Shared Eigenbasis
ZZ	IZ	ZI	Separable
XX	IX	XI	Separable
YY	IY	YI	Separable
XY	ZX	YZ	Entangled
YX	ZY	XZ	Entangled

Table 4.1: MUB for two qubits. For the first 3 bases, the shared eigenbases has fully separable eigenvectors. The last 2 bases have fully entangled eigenvectors.

It is known that for N qubits, there exists a MUB with $2^N + 1$ rows and $2^N - 1$ Pauli strings per row. This is optimal in the sense that $2^N - 1$ is the maximum possible number of distinct Pauli strings (excluding Identity) within a commuting family. In Section 4.5, this result will give us insight into the bounds on our MIN-COMMUTING-PARTITION approach.

4.3 Prior Work

Some of the theoretical aspects of our work were concurrently and independently developed by two other research groups (our work was first presented a month earlier [155]). The four relevant papers, [156] from Waterloo and [157, 158, 159] from Toronto all share with our work a high level goal of reducing the cost of VQE by exploiting the simultaneous measurability of commuting Pauli strings. In particular, [156] maps the measurement cost reduction goal to a graph coloring problem. [157] and [158, 159], which respectively consider Qubit-Wise

Commutativity and General Commutativity (defined in Section 4.4), treat measurement cost reduction as a minimum clique cover problem. The core ideas of these four papers can be compared to Sections 4.4-4.5 and Section 4.12 in this chapter.

Our work is differentiated by a systems perspective that gives explicit attention to the classical computation costs for compilation and transpilation, as well as quantum overheads. The graph algorithms discussed in [156, 157, 158, 159] incur impractical classical costs that may undo potential speedups from simultaneous measurement. We remedy this issue by introducing problem-aware techniques that operate on molecular Hamiltonian graphs in linear time and hence preserve speedups, as discussed in Section 4.6. Also, in Section 4.7, we introduce a synthesis tool for simultaneous measurement circuits, in recognition of the fact that simultaneous measurement does incur a quantum overhead in additional gates and coherence requirements. To the best of our knowledge, this is the first synthesis tool that constructs simultaneous measurement circuits efficiently in both the classical compilation cost and in the quantum circuit complexity. Sections 4.8 and 4.9 present benchmark results and experimental results validating that the classical and quantum costs of simultaneous measurement are worthwhile. Additionally, we study the statistics of simultaneous measurement in Section 4.10 and demonstrate a constructive procedure to guard against corruption from covariance terms.

Prior to this month, strategies for simultaneous measurement in VQE had not been studied formally, aside from the initial suggestion of measurement partitioning in [87]. Most experimental implementations of VQE, for instance [90, 91, 151, 160], did at least perform measurement partitioning on an ad hoc basis, via inspection of the Hamiltonian terms. Inspection is insufficient for larger molecules, because the underlying problem is NP-Hard, as described in Section 4.12. The improvement in these experimental works due to simultaneous measurement is indicated by the reduction from the # Pauli Strings to QWC (Qubit-Wise Commutation) column in Table 4.2. The last column considers General Commutation (GC) partitioning, which we introduce and evaluate in this chapter. Even for the small molecules that have been studied experimentally thus far, GC achieves significant cost reductions over both Naive and QWC partitions.

Molecule	# Pauli Strings	QWC	GC
H_2 [90]	4	2	2
LiH [90]	99	25	9
BeH_2 [90]	164	44	8
H_2 (Bravyi-Kitaev) [151]	5	3	2
H_2 (Jordan-Wigner) [151]	14	5	2
H_2O [91]	21	3	3

Table 4.2: State preparation and measurement costs from prior VQE experiments that performed Pauli string partitioning on an ad hoc basis. # Pauli Strings indicates the number of measurement partitions that would be needed naively. QWC expresses the number of Qubit-Wise Commuting partitions that were actually measured via ad hoc inspection—we propose a more formal partitioning procedure in Section 4.5. GC foreshadows the General Commuting partitions that our techniques described in Sections 4.4.3 and 4.5 - 4.6 achieve.

In software implementations, both the OpenFermion [143] and Rigetti PyQuil [65] libraries were recently augmented with functions for simultaneous measurement via Qubit-Wise Commutation: `group_into_tensor_product_basis_sets()` and `group_experiments()` respectively. However, these software implementations do not consider General Commutativity and suffer from at least N^8 scaling in runtime, which may undo the potential speedup from simultaneous measurement.

An alternative perspective on the reduction of measurement cost in VQE was introduced in [161] which takes the approach of transforming molecular Hamiltonians to *create* commutativity and reduce the number of qubits needed. Another prior paper [162] operates in a related mathematical setting, using feedforward measurements to create QWC (though we note that feedforward measurements are equivalent to standard unitary transformations by the principle of deferred measurement [32]).

Aside from state preparation and measurement costs, recent work has focused on improving other elements of the VQE pipeline. In the classical stage, [129, 87, 163] describe improvements to the classical optimizer and [96, 18] present techniques for optimized pulse-level compilation. At the quantum stage, [152, 164] propose improvements to ansatzes and [87, 57] demonstrate procedures for error mitigation. We note that all of these techniques apply to orthogonal stages of the VQE pipeline and therefore can compose directly on top of our work.

4.4 Analysis of Commutativity

We analyze the commutativity of the terms present in Hamiltonian decompositions. Two terms A and B commute, if their *commutator* is 0:

$$[A, B] := AB - BA = 0 \rightarrow AB = BA$$

As mentioned in Section 4.2.2, two commuting terms are simultaneously diagonalizable by a shared eigenbasis.

In our case, the terms in an N-qubit Hamiltonian are Pauli strings, which are N-fold tensor products of the Pauli matrices,

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Pauli strings are also referred to in other literature as members of the Pauli Group, G_N.

We seek to analyze when two Pauli strings commute. While most of these results are known, they are usually discussed in the context of the stabilizer formalism and quantum error correction. We present the elements relevant to VQE here, with foreshadowing of our key techniques.

4.4.1 Single Qubit Case

First, let's note the commutation relations for single qubit Pauli matrices:

- I commutes with everything else. Specifically, $[I, I] = [I, X] = [I, Y] = [I, Z] = 0$.
- X, Y, and Z commute with themselves. $[X, X] = [Y, Y] = [Z, Z] = 0$.
- The other pairs form a cyclic ordering. In particular, $[X, Y] = iZ$, $[Y, Z] = iX$, $[Z, X] = iY$. Flipping the commutator bracket order negates the result.

4.4.2 Qubit-Wise Commutativity (QWC)

The simplest type of commutativity is Qubit-Wise Commutativity (QWC). Two Pauli strings QWCommute if at each index, the corresponding two Pauli matrices commute. For instance, $\{XX, IX, XI, II\}$ is a QWC partition, because for any pair of Pauli strings, both indices feature commuting Pauli matrices.

As mentioned in Section 4.3, QWC has been leveraged in past experimental work for small molecules [90, 91, 151, 160] by ad hoc inspection of the Hamiltonian terms. However, Section 4.12 demonstrates that optimally partitioning Pauli strings into QWC families is NP-Hard, so an efficient approximation algorithm is needed for larger Hamiltonians with more Pauli strings.

QWC is also referred to in other work as Tensor Product Basis (TPB) [90, 143, 65], recognizing the fact that for a family of QWC Pauli strings, the vectors in the simultaneous eigenbasis can be expressed as a tensor product across each qubit index, with no entanglement. As shown in Section 4.7, this makes simultaneous measurement very easy for QWC partitions.

4.4.3 General Commutativity (GC)

QWC is sufficient but not necessary for commutation between Pauli strings. For example, $\{XX, YY, ZZ\}$ is a commuting family, even though none of the pairs are QWC—at both indices the Pauli matrices always fail to commute. The most general rule for commutation of two Pauli strings is that they must fail to commute at an *even* number of indices—2 in the example of $\{XX, YY, ZZ\}$. We refer to this most general form of commutativity as General Commutativity (GC), and its proof is below. Note that QWC is simply the subset of GC corresponding to the case where the number of non-commuting indices is 0 (which is even).

Theorem 1. *Consider two N-qubit Pauli strings,*

$$A = \bigotimes_{j=1}^{N} A_j \text{ and } B = \bigotimes_{j=1}^{N} B_j$$

where $A_j, B_j \in \{I, X, Y, Z\}$. A and B commute (GC) iff A_j and B_j fail to commute on an even number of indices.

Proof. For Pauli matrices that don't commute, $A_iB_i = -B_iA_i$. Thus, we can write AB as

$$AB = \bigotimes_{j=1}^{N} A_jB_j = \bigotimes_{j=1}^{N} \begin{cases} B_jA_j & \text{if } [A_j, B_j] = 0 \\ -B_jA_j & \text{if } [A_j, B_j] \neq 0 \end{cases} = (-1)^k BA$$

where k is the number of indices where $[A_j, B_j] \neq 0$. For AB to equal BA, we require $(-1)^k = 1$, which requires k to be even. Thus, A and B commute iff A_j and B_j don't commute on an even number of indices. □

Figure 4.3 depicts the commutation relationships between all 16 2-qubit Pauli strings. Edges are drawn between Pauli strings that commute—a blue edge indicates that the pair is QWC and a red edge indicates that the pair is GC-but-not-QWC. The II identity term QWCommutes with every other Pauli string.

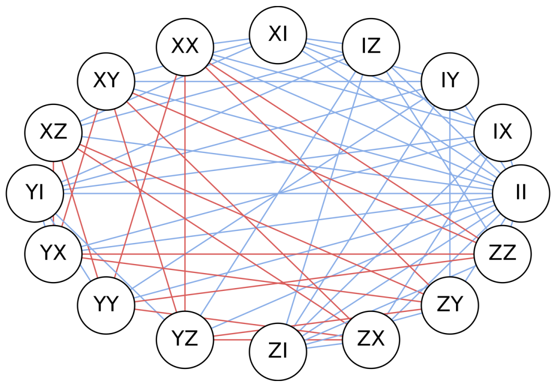


Figure 4.3: This is the commutation graph (also known as a compatibility graph [2]) for all 16 2-qubit Pauli strings. An edge appears when two Pauli strings commute. The blue edges indicate Pauli strings that commute under QWC (which is a subset of GC). The red edges commute under GC-but-not-QWC.

4.5 MIN-CLIQUE-COVER on Hamiltonian

We refer to our core problem of interest as MIN-COMMUTING-PARTITION: given a set of Pauli strings from a Hamiltonian, we seek to partition the strings into commuting families such that the total number of partitions is minimized. While the underlying structure of Pauli matrices and their commutation relationships raises the possibility that MIN-COMMUTING-PARTITION may be efficiently solvable, it turns out to be NP-Hard, as we prove in Section 4.12. Moreover, MIN-COMMUTING-PARTITION is hard even when we only consider the restricted commutativity of QWC. Thus, the ad hoc QWC partitioning techniques from past experimental work [90, 91, 151, 160] are likely to have limited potential for larger molecules.

Instead of solving MIN-COMMUTING-PARTITION exactly, we approximately solve it by mapping to a graph problem as suggestively expressed by the graph representation in Figure 4.3. Observe that cliques (fully connected subgraphs where each pair of Pauli strings commutes) are relevant because all of the strings in a clique can be measured simultaneously. Therefore, we seek the MIN-CLIQUE-COVER, i.e. the smallest possible set of cliques whose union spans all vertices. As an example, Figure 4.4 shows the commutation graph for LiH's

4-qubit Hamiltonian and its MIN-CLIQUE-COVERs using QWC edges and using GC edges.

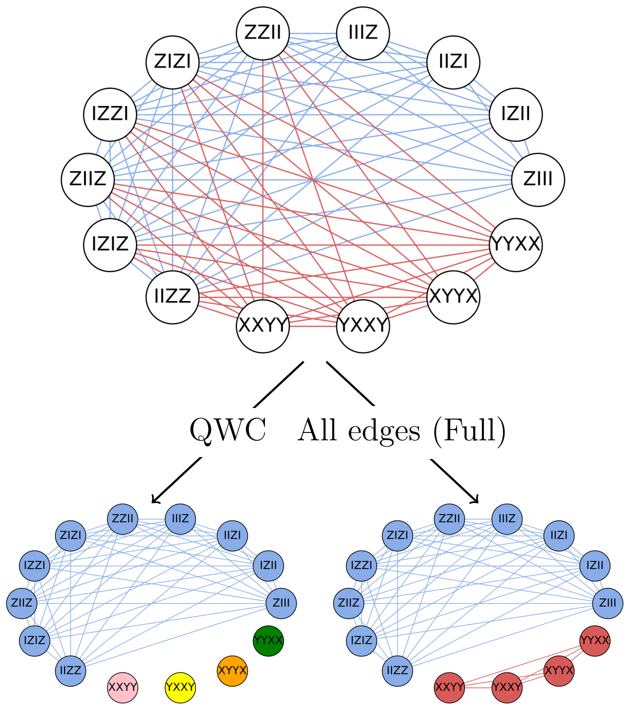


Figure 4.4: The top commutation graph shows both QWC (blue) and GC-but-not-QWC Commuting (red) relationships between the Pauli string's in LiH's Hamiltonian. The vertex colors in the bottom two graphs indicate MIN-CLIQUE-COVERs using only QWC edges (left) or using all edges (right). The reduction in measurement partitions from Naive (measuring each Pauli string separately) to QWC to GC is $14 \rightarrow 5 \rightarrow 2$.

MIN-CLIQUE-COVER, in its decision version, is one of the classic Karp NP-Complete problems [165], so efficiently finding the minimal possible clique cover for a general graph is unlikely. Moreover, finding a guaranteed "good" clique cover approximation is also NP-Hard for general graphs [166]. However, molecular Hamiltonian graphs are highly structured owing both to features of the Pauli commutation graph [167] and to patterns in the Pauli strings that arise in molecular Hamiltonians (we explicitly address and exploit the latter in Section 4.6). This suggests that MIN-CLIQUE-COVER approximation algorithms may yield reasonably good results. Before discussing the approximation algorithms we used, we discuss bounds on the MIN-CLIQUE-COVER and the relationship to whether the partitions are QWC or GC.

4.5.1 Bounds via MUBs

Note that 2^N separate Pauli strings can be measured via a single simultaneous measurement. For instance, consider the 2^N set of Pauli strings of form $(I \text{ or } Z)^{\otimes N}$. All such Pauli strings can be simultaneously measured by simply measuring in the Z basis on each qubit. This example is suggestive of the power of simultaneous measurement. In the graph picture, it means that cliques exist of size 2^N, which means that simultaneous measurement can lead to an exponential reduction in quantum cost relative to Naive separate measurements.

In the case of VQE, we will consider graphs that have only a polynomially sized ($O(N^4)$) number of Pauli strings. It is still enlightening to consider the MIN-CLIQUE-COVER on the N-qubit graph comprising all $4^N - 1$ possible Pauli strings (in this analysis, we exclude $I^{\otimes N}$ which commutes with everything else). Per the MUB formalism introduced in Section 4.2.6 and as suggested in the previous paragraph, a clique of Pauli strings can contain at most $2^N - 1$ vertices. This suggests that at least $2^N + 1$ cliques are needed to cover all $4^N - 1$ possible Pauli strings on N qubits. In fact, this lower bound is exactly attainable—a MUB is exactly such a covering of all N-qubit Pauli strings by disjoint cliques. Again, this illustrates the potential of simultaneous measurement—a square root reduction is achieved in the total number of state preparations and measurements needed to cover all possible N-qubit Pauli strings.

Many of the partitions produced by MUBs have entanglement in the shared eigenbasis: for example, the bottom two rows of the MUB in Table 4.1. This means that the MIN-CLIQUE-COVER corresponding to a MUB requires GC edges and not just QWC edges. Next, we further discuss the advantage of GC over QWC.

4.5.2 QWC vs. GC

GC captures a much denser commutation graph than QWC does, and therefore has more opportunities for larger cliques and thereby smaller clique covers.

We first consider the commutation graph of QWC, over all possible N-qubit Pauli strings; this graph has 4^N vertices. Given a Pauli string with I on k indices, it QWC commutes with exactly $4^k \cdot 2^{N-k} - 1 = 2^{N+k} - 1$ other Pauli strings: on the 'partner' string, the k indices are unrestricted and the $N-k$ indices can either match the original Pauli matrix or be I (we subtract 1 to not count the original Pauli string). Since there are $\binom{N}{k}3^{N-k}$ terms with I on exactly k indices, we see that

$$|E| = \sum_{k=0}^{N} \frac{\binom{N}{k}3^{N-k}(2^{N+k}-1)}{2} = \frac{10^N - 4^N}{2}$$

This corresponds to an asymptotic graph density of

$$\lim_{N\to\infty} \frac{|E|}{|V|(|V|-1)/2} = \lim_{N\to\infty} \frac{(10^4 - 4^N)/2}{4^N(4^N-1)/2} = \lim_{N\to\infty} (5/8)^N = 0.$$

In other words, the QWC graph is extremely sparse. By contrast, the GC graph is dense: consider two random Pauli strings. The indicator variable denoting whether the two strings commute on the ith index is a Bernoulli random variable. Therefore, the GC commutation graph corresponds to when the sum over N such independent variables is even, i.e. when a Binomial random variable is even. Asymptotically, this occurs with $\frac{1}{2}$ probability—thus the asymptotic graph density for GC is $\frac{1}{2}$, much denser than for QWC.

Although GC leads to smaller MIN-CLIQUE-COVERs than QWC, QWC does have cheaper simultaneous measurement circuits, as we will see in Section 4.7. However, the cost of GC simultaneous measurement will still turn out to be favorable, because circuit costs in VQE are dominated by the ansatz preparation.

4.5.3 Approximation Algorithms Tested

In our benchmarking, we performed MIN-CLIQUE-COVERs using the Boppana-Halldórsson algorithm [168] included in the NetworkX Python package [169], as well as the Bron-Kerbosch algorithm [170] which we implemented ourselves. These heuristics approximate a MAX-CLIQUE whose vertices are marked; we then recurse on the residual unmarked graph, repeating until all vertices are marked. We also used the `group_into_tensor_product_basis_sets()` approximation implemented by OpenFermion [143]—this approximation is a non-graph-based randomized algorithm that only finds QWC partitions. Section 4.8 presents results across a range of molecules and Hamiltonian sizes.

While the benchmark results indicate promising performance in terms of finding large partitions, it is critical to also consider the classical computation cost of performing the MIN-CLIQUE-COVER approximation. First, the Bron-Kerbosch algorithm has a worst case exponential runtime. Therefore, its optimality should be interpreted as a soft upper bound on how well other standard approximation algorithms can approximate a MIN-CLIQUE-COVER. The Boppana-Halldórsson algorithm's runtime is polynomial but is not well studied. Our benchmarks and theoretical analysis indicate roughly quadratic scaling in graph size. Some polynomial benchmarks considered in the other concurrent work scale as much as cubically in the graph size.

However, this poses a problem—the Hamiltonian graph has N^4 terms, so a quadratic or cubic runtime in the number of vertices implies N^8 or N^{12} scaling in classical precomputation time. Beyond simply implying impractical scaling rates, these runtime ranges may exceed the quantum invocation cost of VQE, in which case, we'd be better off just running VQE in the Naive fashion. In particular, recall that the UCCSD ansatz has $O(N^3)$ circuit depth after parallelization and that naively, $O(N^4)$ state preparations are needed per ansatz. The total quantum invocation cost of VQE therefore scales as N^7 multiplied by the number of ansatz states explored, though we note that both the ansatz exploration and the naive $O(N^4)$ measurements could be parallelized given multiple quantum machines. The number of ansatz states explored is an open question that depends on the classical optimizer, the ansatz type, and the variational landscape. Nonetheless, we can make rough estimates by noting that the

VQE ansatz has $O(N^4)$ parameters, and rough theoretical results suggest anywhere from $O(N^4)$ iterations under the default SciPy optimization settings [171] to $O(N^{12})$ under matrix inversion techniques. Further work is needed to understand the exact cost of VQE, but there is a strong case that standard graph approximation algorithms may have higher asymptotic cost than simply executing VQE naively without simultaneous measurement optimization. In the case of many expensive MIN-CLIQUE-COVER approximation algorithms, it seems likely that it would be better to simply skip the partitioning step and just measure the Pauli strings naively.

In the next section, we remedy this concern by presenting a MIN-COMMUTING-PARTITION approximation that exploits our knowledge of the structure of molecular Hamiltonians and their encodings into qubits. The resulting approximation algorithm runs in $O(N^4)$ time (linear in the number of Pauli strings, i.e. the graph size), which is safely below the quantum invocation cost of VQE.

4.6 Linear-Time Partitioning

As discussed in the previous section, standard MIN-CLIQUE-COVER approximations may be unsuitable since the classical cost of partitioning can exceed the quantum cost from naively running VQE. This motivates us to inspect features of molecular Hamiltonians and develop a new partitioning strategy accordingly. At a high level, our new strategy is context-aware and attacks the MIN-COMMUTING-PARTITION problem at a different abstraction level, namely the encoding stage from fermionic Hamiltonian to qubit Hamiltonian. By contrast, the previous approximations are unaware of molecular properties.

For convenience, we repeat Equation 4.1 for the molecular Hamiltonians:

$$H = \sum_{p}^{N}\sum_{q}^{N} h_{pq} a_p^\dagger a_q + \sum_{p}^{N}\sum_{q}^{N}\sum_{r}^{N}\sum_{s}^{N} h_{pqrs} a_p^\dagger a_q^\dagger a_r a_s$$

where $a^\dagger$ and a denote raising and lowering operators that act on fermionic modes.

The N^4 scaling of the number of terms in the Hamiltonian is clear from the second summation. In particular, the asymptotically-dominant terms are of form $a_p^\dagger a_q^\dagger a_r a_s$ with $p \neq q \neq r \neq s$. These $O(N^4)$ terms are known as the *double excitation* operators [172]. At the scale of smaller molecules, the $O(N)$ terms of form $a_p^\dagger a_p$ and the $O(N^2)$ terms of form $a_p^\dagger a_q^\dagger a_p a_q$ are frequent. These are termed the *number* and *number-excitation* operators respectively. We will treat both the asymptotically-dominant terms and the frequent-for-small-molecules terms in this section.

The commutation relationships of fermions are different from the commutation relationships of qubits. Thus, an encoding step is needed to convert the fermionic Hamiltonian into a qubit Hamiltonian. We consider the most common [109] such encodings: Jordan-Wigner [146], Parity [147], and Bravyi-Kitaev [148].

4.6.1 Jordan-Wigner

Under the Jordan-Wigner encoding, we make the fermion-to-qubit transformations:

$$a_p \rightarrow \frac{X_p + iY_p}{2} Z_{p-1}...Z_0, \quad a_p^\dagger \rightarrow \frac{X_p - iY_p}{2} Z_{p-1}...Z_0$$

with I on every other index.

Double excitation operators

. For the asymptotically dominant $O(N^4)$ terms of form $a_p^\dagger a_q^\dagger a_r a_s$ (WLOG, $p > q > r > s$), we end up with the 16 Pauli strings matching the regular expression:

$$(X_p|Y_p)Z_{p-1}...Z_{q+1}(X_q|Y_q)(X_r|Y_r)Z_{r-1}...Z_{s+1}(X_s|Y_s)$$

Thus, we see that the Jordan-Wigner transformation turns each of the N^4 fermionic terms into a sum over 16 Pauli strings. Moreover, these 16 Pauli strings are disjoint from the ones generated by a $a_{p'}^\dagger a_{q'}^\dagger a_{r'} a_{s'}$ term. Consider the commutation graph of the 16 Pauli strings. All indices except for p, q, r, and s immediately commute, so the commutativity graph only needs to consider the p, q, r, and s indices. Figure 4.5 depicts the commutation graph, which has a MIN-CLIQUE-COVER of 2. Thus, this yields a strategy for reducing the number of measurement partitions by 8x: we collect all Pauli strings from fermionic terms of form $a_p^\dagger a_q^\dagger a_r a_s$ (and from the 4! permutations of the indices) and measure them using 2 GC partitions instead of 16 Naive partitions.

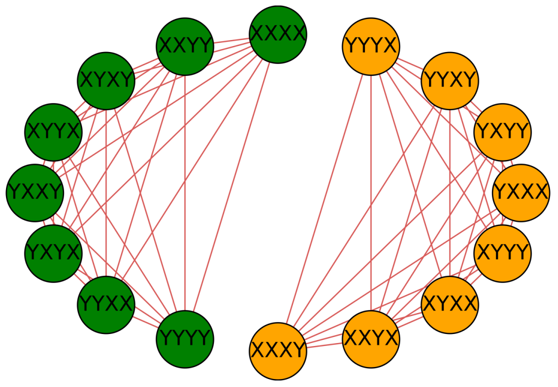


Figure 4.5: The 16 relevant Pauli strings in the Jordan-Wigner encoding of $a_p^\dagger a_q^\dagger a_r a_s$ have a MIN-CLIQUE-COVER of size 2.

For molecular Hamiltonians, we generally expect to have $h_{pqrs} = h_{srqp}$, because of the nature of these calculations via integrals and the fact that electrons are indistinguishable. In this case, only 8 terms arise (as noted in another context by [172]), specifically the green 8-clique in Figure 4.5. Thus again, we can achieve an 8x reduction.

Number and number-excitation operators

While the 8-fold reduction in the partitions of the $O(N^4)$ $pqrs$ terms is the asymptotic bottleneck, we also note a useful reduction for the smaller terms which are significant for smaller molecules.

For the $O(N)$ number operators of form $a_p^\dagger a_p$, multiplying out the Jordan-Wigner encoding yields the Pauli string Z_p. For the $O(N^2)$ number-excitation operators of form $a_p^\dagger a_q^\dagger a_p a_q$, the Jordan-Wigner encoding yields the Pauli string $Z_p Z_q$.

Observe that all of these Pauli strings commute and therefore can be simultaneously measured. Moreover, they are QWC, so the simultaneous measurements are cheap, as we will see in Section 4.7. While this result may appear obvious from inspection of small molecular Hamiltonians, which have many Pauli strings of form $I...IZI...I$, we underscore that it is not obvious to a context-unaware MIN-CLIQUE-COVER approximation.

4.6.2 Parity Encoding

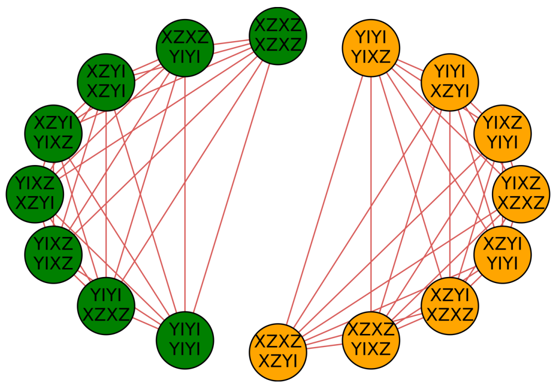


Figure 4.6: Similar to the Jordan-Wigner case, the 16 relevant Pauli strings in the Parity encoding of $a_p^\dagger a_q^\dagger a_r a_s$ have a MIN-CLIQUE-COVER of size 2.

For the Parity encoding, we make the transformations:

$$a_p = X_{N-1}...X_{p+1}\frac{X_p Z_{p-1} + iY_p I_{p-1}}{2}$$

$$a_p^\dagger = X_{N-1}...X_{p+1}\frac{X_p Z_{p-1} - iY_p I_{p-1}}{2}$$

Double excitation operators

WLOG, suppose $p-1>q, q-1>r, r-1>s$. Multiplying out $a_p^\dagger a_q^\dagger a_r a_s$ we see that the parity encoding creates Pauli strings matching the regular expression:

$$(X_p Z_{p-1}|Y_p I_{p-1})X_{p-2}...X_{q+1}(X_q Z_{q-1}|Y_q I_{q-1})...$$

$$...(X_r Z_{r-1}|Y_r I_{r-1})X_{r-2}...X_{s+1}(X_s Z_{s-1}|Y_s I_{s-1})$$

Only indices $p, p-1, q, q-1, r, r-1, s$, and $s-1$ are relevant for commutativity. Once again expanding the resulting 16 Pauli strings, we see that the commutation graph has a MIN-CLIQUE-COVER of size 2, as depicted in Figure 4.6. Thus, we can again achieve an 8x reduction in the number of partitions by performing simultaneous measurement across these indices. However, note that the simultaneous measurement circuit now involves 8 indices, so it will be more expensive than the simultaneous measurement circuit for the Jordan-Wigner encoding.

Number and number-excitation operators

We also again consider the $O(N)$ and $O(N^2)$ operators that are frequent in smaller molecules. The parity encoding on the number and number-excitation operators gives rise to Pauli strings of form $Z_p Z_{p-1}$ and $Z_p Z_{p-1} Z_q Z_{q-1}$ respectively. Again, we see that for small molecules, the parity encoding creates a large set of QWC Pauli strings.

4.6.3 Bravyi-Kitaev

The Bravyi-Kitaev coding is asymptotically favorable for Hamiltonian simulation because it requires asymptotically fewer non-I operators per Pauli string by only selecting a subset of indices to perform partial sums needed in the fermion-to-qubit encoding. As a result, every

a_p or $a_j^\dagger$ term involves a subset of indices ($> p$) that carry the X update, and a subset of the indices ($< j$) that require the phase correction. This complicates the commutation structure of $a_p^\dagger a_q^\dagger a_r a_s$ and there is not an immediately obvious clique cover strategy–we identify this as an open question.

4.7 Circuits for Simultaneous Measurement

Once an approximate MIN-COMMUTING-PARTITION solution has been generated, a natural question arises of how to actually perform the necessary simultaneous measurement for each commuting partition. In the case of Naive partitions where each Pauli string is measured separately, the measurement circuit is trivial. In particular, recall from Section 4.2 that we simply perform the H and $HS^\dagger$ operations on the indices with X or Y respectively, and then we measure every qubit in the Z basis. Thus, we need just $O(N)$ fully-parallelizable single qubit gates; more specifically, we require $k \leq N$ single qubit gates, where k is the number of indices in the Pauli string that equal X or Y.

Simultaneous measurement is also similarly straightforward in the case of QWC partitions. Each index of a QWC partition is characterized by a measurement basis. For example, consider the task of simultaneously measuring the two QWC Pauli strings $XIYIZI$ and $IXIYIZ$. We simply apply H to the left two qubits and $HS^\dagger$ to the right two qubits. The resulting qubits can all be measured in the standard Z basis, and the corresponding outcomes indicate the X, X, Y, Y, Z, and Z outcomes as desired. In terms of circuit cost, QWC measurement is essentially identical to Naive measurement: $O(N)$ single qubit gates are required, and the gates are fully parallelizable to constant depth.

While Naive and QWC partition measurements are straightforward, GC partition measurements are nontrivial. We now introduce a circuit synthesis procedure enabling these measurements, and we analyze both the quantum and classical costs of this procedure. To the best of our knowledge, this is the first work explicitly demonstrating how to perform simultaneous measurement in the general case of GC Pauli strings. We implemented our circuit synthesis tool as a Python library and validated it across a wide range of molecular Hamiltonians.

4.7.1 Background

As discussed in Section 4.2, performing a simultaneous measurement amounts to applying a unitary transformation in which the columns of the unitary matrix are the simultaneous eigenvectors of the commuting Pauli strings in the partition. After applying such a transformation and then performing standard Z-basis measurements, the outcomes are mapped directly to measurements of the Pauli strings of interest. One approach to synthesize a simultaneous measurement circuit would be to explicitly compute the matrix of simultaneous eigenvectors and then apply one of many possible unitary decomposition techniques [173, 174, 175, 176, 177, 178] to this matrix. However, this approach is not sufficient for two reasons. First, in general, decomposition techniques trade off between requiring intractable quantum circuit depth, requiring intractable classical compilation time, and yielding only approximations to the desired transformation. Second, and most importantly, these techniques require us to compute the simultaneous eigenvectors and input them to the decomposer. In general, the simultaneous eigenvectors resulting from GC can be fully entangled across all N indices, and they are represented by a 2^N-sized column vector. The corresponding unitary matrix would be doubly exponentially sized in N, erasing any potential quantum advantage.

With this in mind, it is clear that any decomposition technique must avoid explicitly computing eigenvectors and writing out exponentially sized unitary matrices. Fortunately, the stabilizer formalism—typically applied to quantum error correction—provides us such a mechanism. Before proceeding, we note that our work is built upon the language of stabilizers introduced in [179] and expanded upon in [180]. While these two papers were applied to error correction and quantum simulation, the core techniques also apply to our use case.

Also, [181] and [182] leverage these stabilizer techniques to perform MUB measurements. Our circuit constructions are drawn from these two papers as well as [183], but stem from a different context and end goal.

4.7.2 An Example: $\{XX, YY, ZZ\}$

We begin with a well-known example. Consider the task of trying to simultaneously measure XX, YY, and ZZ, a GC (but not QWC) partition. The simultaneous eigenvectors of these Pauli strings are known as the four *Bell states*:

$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}, \qquad |\Phi^-\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}},$$

$$|\Psi^+\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}, \qquad |\Psi^-\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}$$

These eigenvectors are linearly independent and span all possible 2-qubit states—hence, they are a basis. Unlike the vectors in the standard computational basis of $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, the eigenvectors in the Bell basis feature entanglement between the two qubits. As a result, measurement in the Bell basis requires interaction between the two qubits, unlike the the Naive and QWC measurements described previously. The quantum circuit in Figure 4.7 is a well-known circuit that performs Bell basis measurement, i.e. simultaneous measurement of XX, YY, and ZZ.

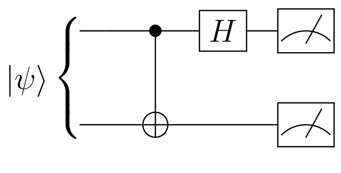


Figure 4.7: Bell basis measurement circuit that simultaneously measures XX, YY, and ZZ on the $|\psi\rangle$ state. After application of these two gates, the measurements of the top and bottom qubits correspond to outcomes for XX and ZZ respectively. The YY outcome is obtained from $YY = -(XX)(ZZ)$.

To understand why this circuit measures XX and ZZ (and also $YY = -(XX)(ZZ)$), we observe that our ultimate goal is to transform a target measurement of $[XX, ZZ]$ into $[ZI, IZ]$—the latter captures the outcomes we actually measure directly via standard Z-basis measurement. An important background result is that after applying some unitary operation U, a target measurement of M on the original state has become equivalent to a measurement of $UMU^\dagger$ [179, 32] on the new state. This is known as *unitary conjugation.*

In the Bell basis measurement circuit, we first apply $U = CNOT$. By computing $UMU^\dagger$ we can see that target measurements of $[XX, ZZ]$ are transformed under conjugation to measurements of

$$[XX, ZZ] \xrightarrow[U=CNOT]{UMU^\dagger} [UXXU^\dagger, UZZU^\dagger] = [XI, IZ].$$

Finally, after applying the Hadamard gate on the top qubit, the measurements are transformed to

$$[XI, IZ] \xrightarrow[U=H\otimes I]{UMU^\dagger} [UXIU^\dagger, UIZU^\dagger] = [ZI, IZ].$$

Thus, this $CNOT$, $H \otimes I$ gate sequence performs the desired transformation of rotating a measurement of $[XX, ZZ]$ into the computational basis, $[ZI, IZ]$. The ordering of the elements is important and indicates that measurement of the top qubit (ZI) corresponds to the XX outcome and measurement of the bottom qubit (IZ) corresponds to the ZZ outcome. As mentioned previously, YY follows as $-(XX)(ZZ)$.

4.7.3 Stabilizer Matrices

In order to consider the general case, we now switch to the formalism of stabilizer matrices. Our notation and terminology is similar to previous work [179, 180, 181, 182, 183], with some deviations for clarity. Within the stabilizer formalism, every N-qubit Pauli string maps to a $2N$-entry column vector. The top N entries indicate whether each corresponding index 'contains' a Z. The bottom N entries correspond to X's. The Y Pauli matrix corresponds to having a 1 in both the Z and X entries, since $Y = iZX$. The stabilizer matrix for a list of Pauli strings is simply the concatenation of the column vectors. As an instructive example, the stabilizer matrix for $[XXX, YYY, ZZZ, XYZ]$ is:

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

For convenience and clarity, we will refer to the top N rows as the *Z-matrix* and the bottom N rows as the *X-matrix.* Recall that our goal is to transform a target set of Pauli strings for simultaneous measurement into the computational basis measurements, $[ZII...I, IZI...I, ..., III...Z]$. We see that the stabilizer matrix for this computational basis simply has an $N \times N$ Identity as the Z-matrix and all zeroes in the X-matrix.

We now seek a procedure to transform the target stabilizer matrix into this computational basis stabilizer matrix. To see how to accomplish such a transformation, it is useful to know unitary conjugation relationships for a basic gate set. Table 4.3 and Table 4.4 list the unitary conjugations of important Pauli strings for 1- and 2- qubit unitary gates respectively.

	$UZU^\dagger$	$UXU^\dagger$
$U = H$	X	Z
$U = S$	Z	Y

Table 4.3: Result of conjugation of Z and X by single qubit gates $U = H$ or S. Note that H can be thought of as a "NOT gate" between X and Z. The S (phase) gate does not affect Z, but does transform X into Y.

	$UZIU^\dagger$	$UIZU^\dagger$	$UXIU^\dagger$	$UIXU^\dagger$
$U = CNOT$	ZI	ZZ	XX	IX
$U = CZ$	ZI	IZ	XZ	ZX
$U = SWAP$	ZI	ZI	IX	XI

Table 4.4: Result of conjugation of ZI, IZ, XI, or IX by two qubit gates $U = CNOT$, CZ, or $SWAP$.

Based on these tables, we can interpret the action of each of these unitaries on a stabilizer matrix. These rules can be verified directly from the tables and are also explained in [180, 182].

- H on the ith qubit swaps the ith and $i+N$th row of the stabilizer matrix (i.e. swaps between corresponding rows of the Z- and X- matrices). It is helpful to think of H as a "NOT gate" that flips Z and X measurements.

- S on the ith qubit sets the (i,i) diagonal entry in the Z-matrix to 0.

- $CNOT$ controlled on ith qubit and targeted on the jth qubit adds the jth row to the ith row and adds $i+N$th row to the $j+N$th row. All additions are performed modulo 2.

- CZ between the i and jth qubits sets the (i,j) and (j,i) symmetric off-diagonal entries of the Z-matrix to 0.

- $SWAP$ between the i and jth qubits swaps the i and jth rows of both the Z and X matrices. This can be seen from the fact that $SWAP = (CNOT)(NOTC)(CNOT)$ and two rows can be swapped with three alternating binary additions.

4.7.4 Circuit Synthesis Procedure

We now have the tools we need for circuit synthesis, which amounts to transforming the stabilizer matrix for a commuting family of Pauli strings into the computational basis stabilizer matrix (which has Identity for the Z-matrix and zeros for the X-matrix). For simplicity, we describe the procedure for the case when the partition of N-qubit Pauli strings is complete and contains N linearly independent elements. This is the hardest case—if the partition is incomplete, the measurement procedure is similar but has more slack, because at least 1 of the qubits will not need to be measured.

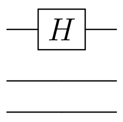
Algorithm 3: Circuit synthesis for sim. measurement

```
input  : $\{P_i\}$, a complete GC family of Pauli strings
output : Circuit for simultaneous measurement of $\{P_i\}$
$M \in F_2^{2N \times N} \leftarrow$ basis of $\{P_i\}$
Full-rankify $X$-matrix by applying $H$ gates
Gaussian eliminate $X$-matrix using CNOT & SWAP gates
for each diagonal element in $Z$-matrix do
    if element is 1 then apply $S$ to corresponding qubit
end
for each element below diagonal of $Z$-matrix do
    if element is 1 then apply $CZ$ to the row-col qubits
end
Apply $H$ to each qubit
Measure each qubit
```

The circuit synthesis procedure is described in Algorithm 3. To develop its intuition, we demonstrate its application to the problem of simultaneously measuring $[IYX, ZZZ, XIX, ZXY]$, which is a GC (but not QWC) family. We initialize the algorithm by setting the stabilizer matrix to a basis of this partition. Note that the fourth term is linearly dependent on the first three, so we exclude it to yield such a basis; in general, we use Gaussian elimination to perform this distillation of the Pauli strings into a basis. The stabilizer matrix for this resulting list of Pauli strings, $[IYX, ZZZ, XIX]$, is:

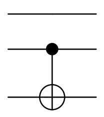
$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

The first step of the simultaneous measurement circuit synthesis is to apply H gates as needed to transform the X-matrix to have full rank (it is currently only rank 2). Such a transformation is always possible and can be found efficiently by Gaussian elimination [180, Lemma 6]. In this case, applying H to the first qubit swaps the first and fourth rows of the stabilizer matrix, yielding an X-matrix of full rank 3:



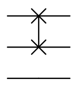
$$\rightarrow \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

Now that the X-matrix is of full rank, we can apply standard Gaussian elimination to row reduce it into the Identity matrix. The CNOT and SWAP gates give us the elementary row operations needed: add one row to another and swap rows. In this example, the X-matrix can be row reduced to the identity by first adding its second row to the third row, and then swapping the first and second rows. Breaking this down, we first observe the effect of the CNOT on the stabilizer matrix:



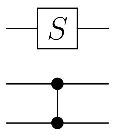
$$\rightarrow \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

And finally the SWAP completes the row reduction, leaving the X-matrix as the identity:



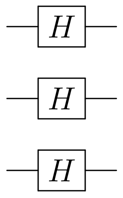
$$\rightarrow \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Notice that the CNOT and SWAP also affected the Z-matrix, which is now a symmetric matrix; this is guaranteed to occur [182]. Now our desired transformation is almost complete. The on-diagonal 1 is erased with S on the first qubit, and the two off-diagonal 1s are erased with a CZ between the second and third qubits. These two operations have no effect on the X-matrix:



$$\rightarrow \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

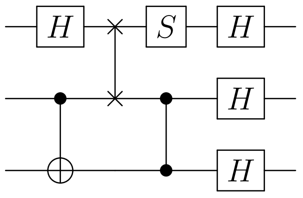
Finally, we apply an H to each qubit, which swaps the Z- and X- matrices, leaving us in the computational basis stabilizer matrix, as desired:



$$\rightarrow \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

The full circuit and resulting transformation is shown below:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} \rightarrow$$



$$\rightarrow \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

4.7.5 Circuit Complexity

The efficiency of Algorithm 3 and the overarching stabilizer formalism stems from the fact that the stabilizer matrices are of size $2N \times N$, and all manipulations are on this tractably-sized matrix. This averts the exponential cost that manipulating simultaneous eigenvectors would entail. In terms of classical cost, the synthesis tool is fast because its slowest step is the Gaussian elimination, which has time complexity of $O(N^3)$ [184].

The actual circuit produced by the synthesis procedure requires only $O(N^2)$ gates in the worst case, as also noted in related results [182, 183]. This follows because the Gaussian elimination can require $O(N^2)$ elementary row operations, which entails $O(N^2)$ CNOT gates. The erasure of off-diagonal elements in the Z-matrix also requires $O(N^2)$ CZ gates.

While the $O(N^2)$ gate count for GC measurement is worse scaling than the $O(N)$ gate count for Naive or QWC measurement, we emphasize that the measurement circuit is preceded by an ansatz preparation circuit that dominates gate counts and depth. In particular, the UCCSD ansatz has $O(N^4)$ gate count and $O(N^3)$ depth after parallelization. Therefore, the cost of simultaneous measurement is asymptotically insignificant. As discussed, we base our studies on UCCSD because the Coupled Cluster approach is the gold standard for quantum computational chemistry [109, 115]. Moreover, UCCSD has shown experimental and theoretical promise, unlike hardware-driven ansatz, which were shown to suffer from "barren plateaus" in the optimization landscape [149, 109]. Even in the case of other non-hardware-driven ansatzes, gate counts and depths generally scale at least as N^3 in order to achieve high accuracy. Thus, the quadratic cost of GC measurement appears to be benign.

We also underscore that the $O(N^2)$ gate count scaling of simultaneous measurement is a worst case scenario, where our partition is dominated by GC-but-not-QWC edges. In practice, this is not the case and we see QWC on many, if not most indices. For example, in the linear-time MIN-COMMUTING-PARTITION 8x approximations presented in Section 4.6 only a constant (4 or 8) number of Pauli string indices have a GC-but-not-QWC relationship in the simultaneous measurements. The remaining $N-4$ or $N-8$ Pauli string indices are QWC. Thus, under this MIN-COMMUTING-PARTITION approximation, the simultaneous measurement circuit gate count is still $O(N)$ and the depth is still parallelizable to $O(1)$.

For reference, we show in Figure 4.8 the simultaneous measurement circuit for the 4 GC-but-not-QWC qubits in the Pauli partition for the Jordan-Wigner transformation. Specifically, this measurement circuit is used to measure the green 8-clique in Figure 4.5. The other $N-4$ qubits are QWC and require single-qubit gates for measurement—this is why the simultaneous measurement gate complexity is still just $O(N)$.

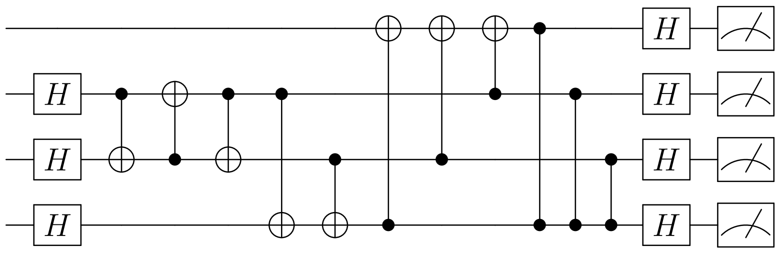


Figure 4.8: Simultaneous measurement circuit generated by our software for the green 8-clique in Figure 4.8. It transforms the measurements of $XXXX, XXYY, XYXY, YXXY$ (which is a basis for the Pauli strings in the green 8-clique) to measurements of $ZIII, IZII, IIZI, IIIZ$.

4.7.6 Measurement Circuit Optimizations

While the circuit synthesis procedure in Algorithm 3 yields a correct simultaneous measurement circuit, it is not necessarily the most optimal circuit possible. For instance, in Figure 4.8, the $SWAP$ (implemented as 3 $CNOT$s) between qubits 2 and 3 can be omitted from the circuit and instead implemented by swapping their subsequent gates, and then accounting for the SWAP classically after the measurements are performed. In other words, the SWAPs in our circuit constructions can be accomplished by simple classical re-labeling of qubit indices.

We also observe that many gates can be parallelized. For example, the depth of Figure 4.8 can be reduced by parallelizing the execution of the CZ gates with the execution of the $CNOT$ gates.

4.8 Benchmark Results

We tested the performance of our simultaneous measurement strategies in Section 4.5 on multiple molecular benchmarks, whose Hamiltonians we obtained via OpenFermion [143]. Our benchmark results encompass both the reduction in number of partitions relative to Naive, as well as the classical computation runtime required to produce the partitioning.

As mentioned, in Section 4.5, the Bron-Kerbosch based MIN-CLIQUE-COVER approximation has exponential worst case runtime and should thus be considered a soft bound on the optimality of partitions produced by other graph approximation algorithms. Figure 4.9 indicates the performance of Bron-Kerbosch in terms of number of commuting partitions (cliques) found using both QWC and GC edges, in comparison to the Naive VQE implementation in which each Pauli string is in a singleton partition. The improvement from Naive to QWC is consistently about 4-5x—a significant reduction especially considering that QWC measurement is cheap. The improvement from Naive to GC ranges from 7x to 12x from H_2 to CH_4 (methane). This suggests that the state preparation cost reduction factor from GC partitioning improves for larger molecules.

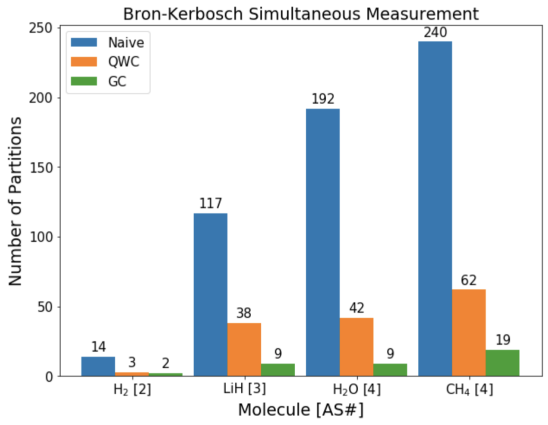


Figure 4.9: Number of QWC and GC partitions (which we are attempting to *minimize*) generated by Bron-Kerbosch for four representative molecules. AS# indicates the number of active spaces for the molecular Hamiltonian.

Figure 4.10 and Figure 4.11 examine partitioning efficacy when we vary the qubit encodings and the number of active spaces considered for the H_2 molecule. Across the qubit encodings, performance is roughly consistent with a 3x improvement from QWC partitions and a 10x improvement from GC partitions. We do note one outlier in that the performance is particularly promising for the Brayvi-Kitaev Super-Fast encoding [148], which achieves a 20x reduction in the number of partitions from Naive to GC. Across the varying active spaces, we again see evidence that the GC partitioning advantage scales with Hamiltonian size, ranging from 3x to 12x as the number of active spaces is increased. This is important and encouraging, because prior work demonstrated that a relatively large number of active spaces are needed to achieve chemical accuracy [116].

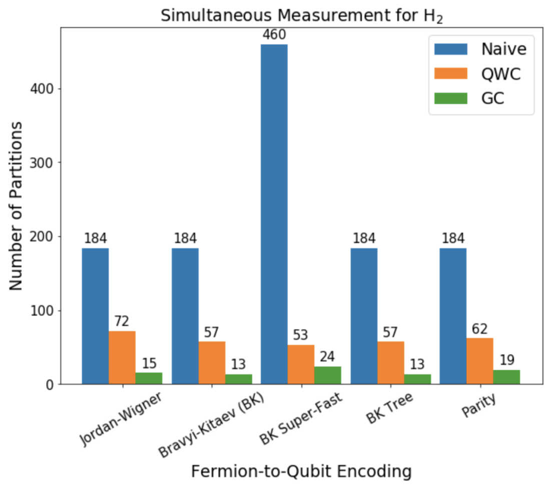


Figure 4.10: Number of QWC and GC partitions generated by Bron-Kerbosch for the H_2 molecule, under different fermion-to-qubit encodings.

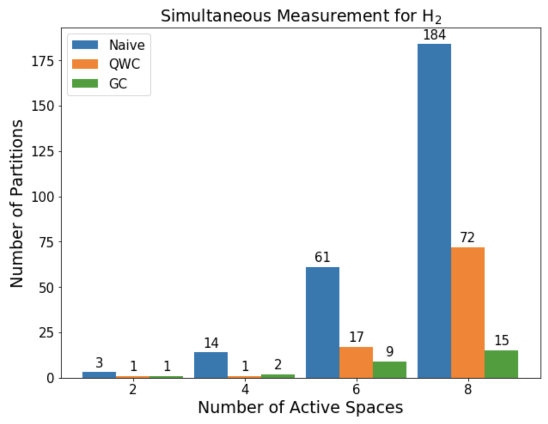


Figure 4.11: Number of QWC and GC partitions generated by Bron-Kerbosch for the H_2 molecule, under different numbers of active spaces.

Along with the Bron-Kerbosch approximations as a loose upper bound on the expected partitioning optimality, we also benchmarked another MIN-CLIQUE-COVER approximation: the Boppana-Halldórsson algorithm, applied to both QWC- and GC- edge graphs. In addition, we also benchmarked with the QWC partitioning heuristic provided by the OpenFermion electronic structure package. We tested each of these algorithms on problem sizes ranging from 4 to 5237 terms in the molecular Hamiltonian. These Hamiltonians correspond to the

H_2, LiH, H_2O, and CH_4 molecules with varying numbers of active spaces. We recorded both the number of partitions generated and the runtime for each algorithm-benchmark pair. Figure 4.12 shows the number of partitions generated for Hamiltonians with up to 5237 Pauli strings. Note that some of the benchmarks were unable to be run due to prohibitive runtime costs on the order of days (e.g. Bron-Kerbosch for $|H| > 1519$ Pauli strings). Figure 4.13 shows a zoom-in for molecules with up to 630 Pauli strings; the y-axis now shows the reduction factor in number of partitions. The plots generally align with our expectations: GC leads to much more optimal partitioning than QWC (recall the arguments in Section 4.5.2, and Bron-Kerbosch GC achieves the fewest number of partitions generated although Boppana-Halldórsson GC has comparable optimality. Among the QWC methods, we consistently see 3-4x reductions in number of partitions over Naive separate measurements, and our Boppana-Halldórsson QWC algorithm marginally outperforms the OpenFermion heuristic.

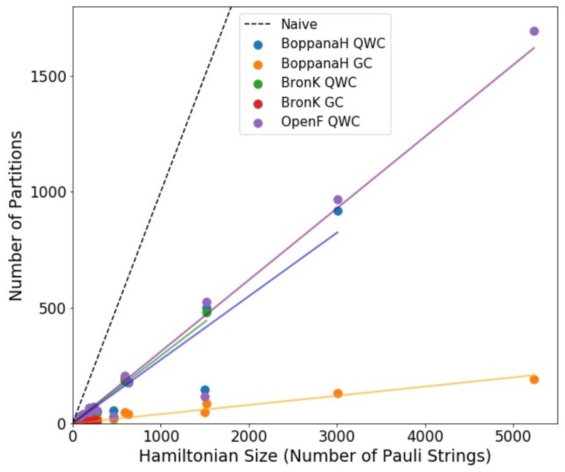


Figure 4.12: Number of partitions found for each algorithm-benchmark pair. Under Naive measurement, the number of partitions would exactly equal the Hamiltonian size (number of Pauli strings). Thus, these techniques all achieve a 4-20x reduction in state preparations and measurements relative to the Naive strategy.

Figure 4.14 plots the wall clock runtimes for each of the algorithm-benchmark pairs; Figure 4.15 focuses on the 0 – 630 Hamiltonian size range. These plots corroborate the exponential worst-case scaling of Bron-Kerbosch and suggest quadratic runtime scaling

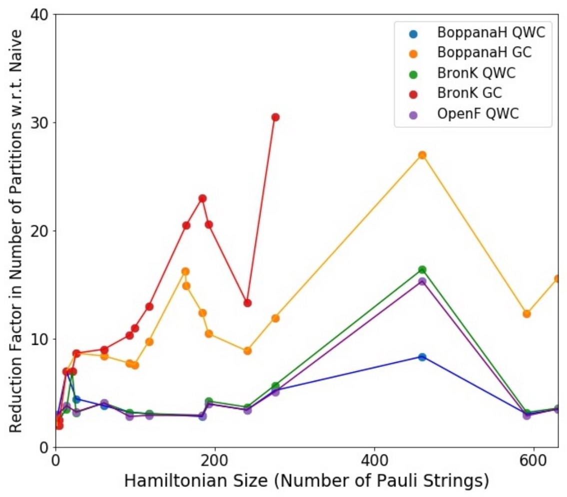


Figure 4.13: Factor of improvement (which we are attempting to *maximize*) over Naive for each of the algorithms benchmarked for Hamiltonian sizes up to 630 terms.

for the Boppana-Halldósson algorithm. OpenFermion's function is clearly the fastest of the algorithms explored, but is also consistently the worst approximation to the MIN-COMMUTING-PARTITION.

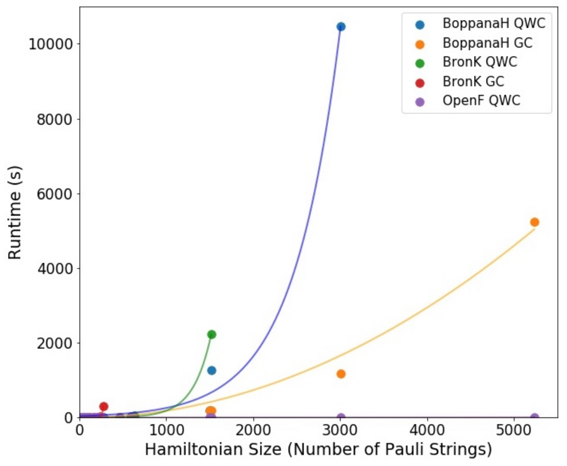


Figure 4.14: Classical computer runtimes for each partitioning algorithm + benchmark pair. Bron-Kerbosch has exponential and Boppana-Halldósson has quadratic runtime scaling. This partitioning step runs as a compilation procedure before the actual quantum invocations of VQE.

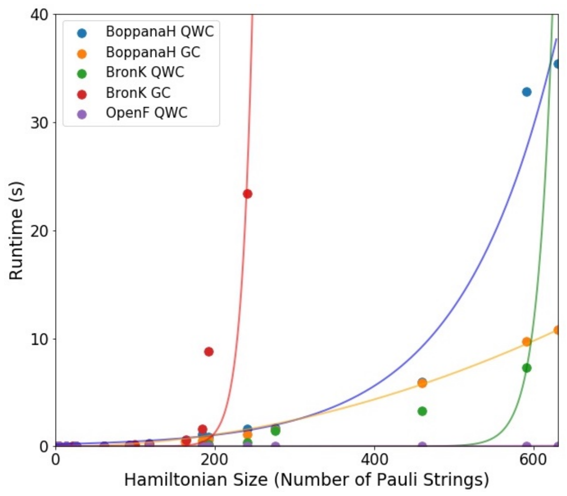


Figure 4.15: Zoom-in of Figure 4.14 for Hamiltonian sizes up to 630 terms.

4.9 Experimental Results

We validated our techniques with a proof of concept demonstration by experimentally replicating a recent result [185]: ground state energy estimation of deuteron, the nucleus of an uncommon isotope of hydrogen. We performed our experiments via the IBM Q Tokyo 20-qubit quantum computer [5], which is cloud accessible.

Following [185], deuteron can be modeled with a 2-qubit Hamiltonian spanning 4 Pauli strings[1]: IZ, ZI, XX, and YY. Under Naive measurement, each Pauli string is measured in a separate partition. Under GC, we can partition into just two commuting families: $\{ZI, IZ\}$ and $\{XX, YY\}$. Recall that the former partition is QWC and can be measured with simple computational basis measurements. The latter partition can be measured by the Bell basis measurement circuit in Figure 4.7.

To establish a fair comparison between Naive measurement and simultaneous measurement we performed experiments in which both settings were allocated an equal budget in total number of shots (trials) allowed. We first considered a resource-constrained setting with a budget of 100 total shots. This corresponds to 25 shots per partition in Naive measurement

1. There is also an II term, but this doesn't actually require any measurement—it just adds a constant offset to the Hamiltonian.

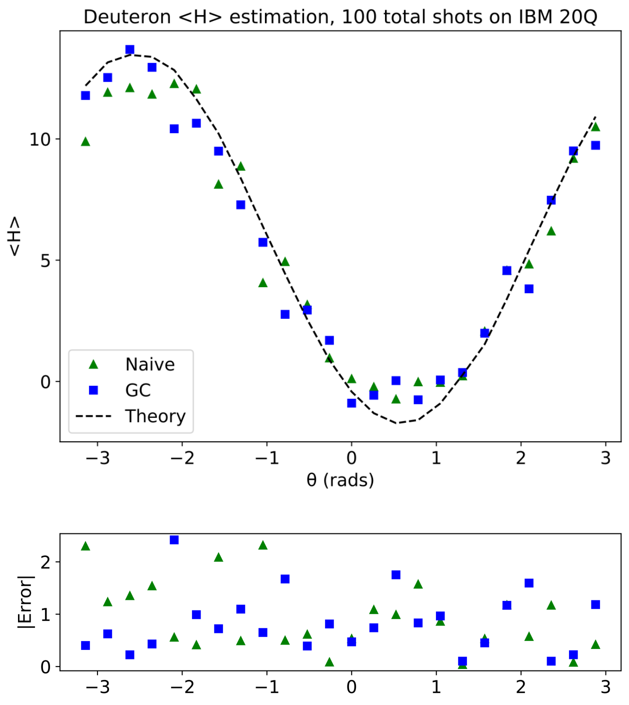


Figure 4.16: Deuteron energy estimation under Naive and GC partitions, as executed on IBM Q20 with a total shot budget of 100. The energies are in MeV. Average error is 11% lower with GC simultaneous measurement than with Naive separate measurements.

and 50 shots per partition in GC simultaneous measurement. Figure 4.16 plots our results for a simplified Unitary Coupled Cluster ansatz with a single parameter and just three gates (two single qubit rotations and one CNOT), as described in [185].

The results indicate reasonable agreement between Naive measurement, GC measurement, and the true (Theory) values. The deviation from Theory stems both from statistical variance due to the low shot budget, as well as systematic noise in the quantum processes. As Figure 4.16's lower |Error| plot indicates, for 13 of the 24 values swept across the θ range, GC measurement had lower error than Naive measurement. On average, the GC measurements had an error of 835 KeV—11% less than the average error of 940 KeV for Naive measurement.

We also ran another experiment with a much higher total shot budget of 4000 (i.e. 1000 shots per partition in Naive and 2000 for GC). In this regime, errors due to systematic quantum noise should dominate over errors from statistical variation. We expect GC simultaneous

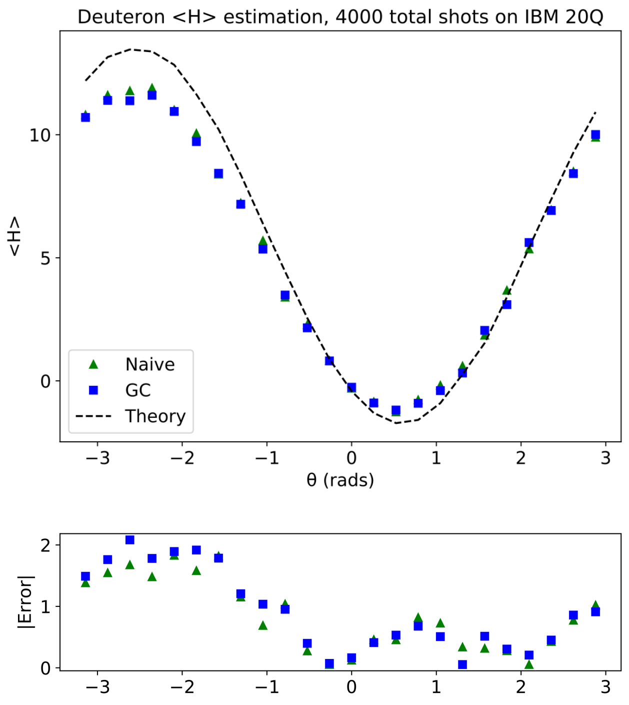


Figure 4.17: Deuteron energy estimation under Naive and GC partitions, as executed on IBM Q20 with a total shot budget of 4000. The energies are in MeV. Average error is 7% lower with Naive separate measurements than with GC simultaneous measurements.

measurement to exhibit more systematic noise because it requires an extra CNOT gate as per the Bell measurement circuit in Figure 4.7. Therefore, we expect better results from Naive measurement than from GC simultaneous measurement. Figure 4.17 plots the experimental results.

For 17 of the 24 values swept across the θ range, Naive measurement does indeed outperform GC simultaneous measurement in terms of lower error. The respective average errors are 848 KeV and 914 KeV, indicating a 7% higher accuracy with Naive measurement.

These results are presented as proof-of-concept that simultaneous measurement achieves higher accuracy when the shot budget is limited. Equivalently, we can achieve equal accuracy with fewer shots (i.e. fewer state preparations) when the shot budget is limited. For several reasons, we note that these experimental results *underestimate* the potential of simultaneous measurement, especially as higher quantum volume devices emerge. In particular:

- the Unitary Coupled Cluster ansatz of [185] is highly simplified and does not yet exhibit the asymptotic $O(N^4)$ scaling. Our argument that simultaneous measurement is cheap hinges on the comparison between $O(N^4)$ ansatz gate count and $O(N^2)$ simultaneous measurement gate count. For this simplified ansatz and small N, simultaneous measurement essentially doubled the gate count. As lower-error devices emerge with the ability to support the full UCCSD ansatz gate count and larger qubit count N, simultaneous measurement circuits will become a negligible cost.

- For a small Hamiltonian like the one considered here, the partitioning gain from GC is only 2x. As indicated in the benchmark results in Section 4.8, we expect up to 30x gains for larger Hamiltonians and possibly a gain factor that continues to linearly increase for larger molecules, based on extrapolation of the benchmark results.

- For current machines, the number of jobs is far more costly than the number of shots for practical purposes, since executions are scheduled at the granularity of jobs. In our executions, we saw this as an immediate and practical advantage of simultaneous measurement. Our total latency was dominated by the number of jobs rather than the number of shots, so our simultaneous measurement results were collected much more rapidly than Naive measurement results, even though both settings had equal total shot budgets.

We re-iterate that these results should only be interpreted as a proof of concept. As machines improve, we expect to see dramatically better results, for the aforementioned reasons.

4.10 Statistics of Simultaneous Measurement: Guarding Against Covariances

We have now shown both how to approximate a MIN-COMMUTING-PARTITION and how to actually construct the requisite simultaneous measurement circuits. Finally, we now address an important question regarding the statistics of simultaneous measurement. This question was first raised by [87, Section IV B2] which proved that simultaneous measurement can actually *underperform* separate measurements due to the presence of covariance terms. In particular, while simultaneous measurement does not bias the estimate $\widehat{\langle H\rangle}$, it can increase the variance of the estimator, relative to separate measurements.

In this section, we first show a specific example from [87] in which simultaneous measurement is *suboptimal*. Then, we prove that such examples are atypical and that the MIN-COMMUTING-PARTITION is still optimal when we have no prior on the ansatz state. Finally, we demonstrate an adaptive strategy for detecting and correcting course in the atypical case when a simultaneous measurement should be split into separate measurements.

4.10.1 An Example

Consider the Hamiltonian, $H = IZ + ZI - XX - YY + ZZ$, following the example of [87]. The commutation graph has a bowtie shape. Figure 4.18 depicts two possible clique partitionings with $k = 2$ and $k = 3$ commuting-family partitions respectively.

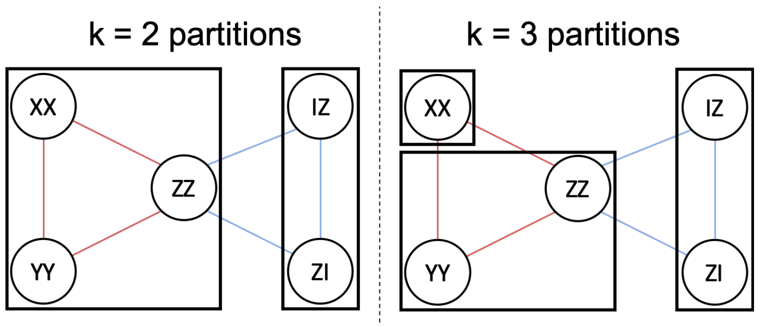


Figure 4.18: Commuting-family partitions of $H = IZ + ZI - XX - YY + ZZ$ with $k = 2$ and $k = 3$.

Thus far, we have worked under the assumption that estimating $\langle H \rangle$ is more efficient with simultaneous measurement than with separate measurements and we have therefore targeted MIN-COMMUTING-PARTITIONs. However, consider a case in which the ansatz state is $|01\rangle$, for the previously stated Hamiltonian.

Since the outcomes of our measurements are random, we quantify the uncertainty around our estimate of the expectation value by $Var(\langle H \rangle)$. Our end goal is to determine the expected value of the Hamiltonian to a target accuracy level ϵ. The expected number of state preparations, n_{expect}, needed to achieve this accuracy for a k-way partitioning is [87]:

$$n_{\text{expect}} = \frac{k \sum_{i=1}^{k} Var(\text{Partition } i)}{\epsilon^2} \tag{4.2}$$

The variance from each partition can be computed from the formula for the variance of a sum of terms:

$$Var(\{\sum_{i=1}^{n} M_i\}) = \sum_{i=1}^{n} Var(M_i) + 2 \sum_{1 \leq i < j \leq n} Cov(M_i, M_j)$$

where $Cov(M_1, M_2) = \langle M_1 M_2 \rangle - \langle M_1 \rangle \langle M_2 \rangle$ and $Var(M) = Cov(M, M)$.

In our case with $|\psi\rangle = |01\rangle$, the primitives evaluate to: $Var(IZ) = Var(ZI) = Var(ZZ) = 0$ and $Var(-XX) = Var(-YY) = 1$. All covariances are 0 except for

$Cov(-XX, -YY) = 1$.

For the $k = 2$ partitioning, we have

$$\begin{aligned}
n_{\text{expect}} &= \\
\frac{2\left[Var(\{-XX, -YY, ZZ\}) + Var(\{ZI, IZ\})\right]}{\epsilon^2} &= \\
2\Big[Var(-XX) + Var(-YY) + Var(ZZ) &+ \\
2Cov(-XX, -YY) + 2Cov(-XX, ZZ) + 2Cov(-YY, ZZ) &+ \\
Var(ZI) + Var(IZ) + 2Cov(IZ, ZI)\Big]/\epsilon^2 & \\
&= \boxed{8/\epsilon^2}
\end{aligned}$$

For the $k = 3$ partitioning, we have:

$$\begin{aligned}
n_{\text{expect}} &= \\
\frac{3\big[Var(\{-XX\}) + Var(\{-YY, ZZ\}) + Var(\{IZ, ZI\})\big]}{\epsilon^2} &= \\
= 3\Big[Var(-XX) + Var(-YY) + Var(ZZ) &+ \\
2Cov(-YY, ZZ) + Var(ZI) + Var(IZ) + 2Cov(IZ, ZI)\Big]/\epsilon^2 & \\
&= \boxed{6/\epsilon^2}
\end{aligned}$$

Thus, due to the contribution of positive covariance between $-XX$ and $-YY$, the $k = 3$ partitioning is better than the $k = 2$ partitioning for this $(H, |\psi\rangle)$ combination.

This phenomenon motivates us to pay close attention to covariances within each partitioning. The worst case scenario is that we end up with positive covariances within each partition. In a best case scenario, we'll have negative covariances within each partitioning, which could dramatically reduce the number of state preparations needed to achieve some desired error on $\langle H\rangle$.

4.10.2 Typical Case

We now observe that examples such as the previous one, in which the MIN-COMMUTING-PARTITION is suboptimal, are atypical. Below, we prove that when we have no prior on the ansatz state $|\psi\rangle$, the expected covariance between two commuting Pauli strings is 0. This validates the general goal of finding the MIN-COMMUTING-PARTITION, because under 0 covariances, the only strategy for reducing n_{expect} in Equation 4.2 is to minimize the total number of partitions k.

Theorem 2. *Given M_1, M_2, two commuting but non-identical Pauli strings, $\mathbb{E}[Cov(M_1, M_2)] = 0$ where the expectation is taken over a uniform distribution over all possible state vectors (the Haar distribution [186, 187]).*

Proof. We consider the following two exhaustive cases:

1. Either M_1 or M_2 is I. WLOG, suppose $M_1 = I$. Then, $Cov(M_1, M_2) = \langle I \cdot M_2\rangle - \langle I\rangle \langle M_2\rangle = 0$.

2. Neither M_1 nor M_2 is I. Since M_1 and M_2 are Pauli strings which have only $+1$ and -1 eigenvalues, the eigenspace can be split into $M_1, M_2 = (-1,-1)$, $(-1,+1)$, $(+1,-1)$, and $(+1,+1)$ subspaces. Moreover, these subspaces are equally sized (proof follows from stabilizer formalism [32, Chapter 10.5.1]). Let us write $|\psi\rangle$ as a sum over projections into these subspaces:

$$|\psi\rangle = a\,|\psi_{-1,-1}\rangle + b\,|\psi_{-1,+1}\rangle + c\,|\psi_{+1,-1}\rangle + d\,|\psi_{+1,+1}\rangle$$

   Under this state, the covariance is $Cov(M_1, M_2)_{|\psi\rangle} = \langle M_1 M_2\rangle - \langle M_1\rangle \langle M_2\rangle = (|a|^2 - |b|^2 - |c|^2 + |d|^2) - (-|a|^2 - |b|^2 + |c|^2 + |d|^2)(-|a|^2 + |b|^2 - |c|^2 + |d|^2)$.

   Now consider the matching state:

$$|\psi'\rangle = b\,|\psi_{-1,-1}\rangle + a\,|\psi_{-1,+1}\rangle + d\,|\psi_{+1,-1}\rangle + c\,|\psi_{+1,+1}\rangle$$

Under $|\psi'\rangle$, the covariance is $Cov(M_1, M_2)_{|\psi'\rangle} = \langle M_1 M_2 \rangle - \langle M_1 \rangle \langle M_2 \rangle = (|b|^2 - |a|^2 - |d|^2 + |c|^2) - (-|b|^2 - |a|^2 + |d|^2 + |c|^2)(-|b|^2 + |a|^2 - |d|^2 + |c|^2)$.

Thus, $Cov(M_1, M_2)_{|\psi\rangle} = -Cov(M_1, M_2)_{|\psi'\rangle}$. Since each $|\psi\rangle$ is matched by this symmetric $|\psi'\rangle$ state, and our expectation is over a uniform distribution of all possible state vectors, we conclude that $\mathbb{E}[Cov(M_1, M_2)] = 0$.

□

4.10.3 Mitigating Covariances: Partition Splitting

While we have now secured the top level goal of initially performing measurements under the MIN-COMMUTING-PARTITION approximation, it is still important to detect and correct course if covariances do turn out to harm our measurement statistics. We now introduce such a strategy that adaptively splits partitions to mitigate harmful covariances.

Our strategy is based on building sample covariance matrices of commuting Pauli strings. If M_1, M_2, and M_3 are Pauli strings, recall that the covariance matrix, $Cov([M_1, M_2, M_3])$, under a fixed state is expressed as follows:

$$\begin{pmatrix} Var(M_1) & Cov(M_1, M_2) & Cov(M_1, M_3) \\ Cov(M_2, M_1) & Var(M_2) & Cov(M_2, M_3) \\ Cov(M_3, M_1) & Cov(M_3, M_2) & Var(M_3) \end{pmatrix}$$

Or, in shorthand notation, where $Var(M_1) = \sigma^2_{M_1}$ and $Cov(M_1, M_2) = \sigma_{M_1 M_2}$:

$$\begin{pmatrix} \sigma^2_{M_1} & \sigma_{M_1 M_2} & \sigma_{M_1 M_3} \\ \sigma_{M_2 M_1} & \sigma^2_{M_2} & \sigma_{M_2 M_3} \\ \sigma_{M_3 M_1} & \sigma_{M_3 M_2} & \sigma^2_{M_3} \end{pmatrix}$$

Note that for commuting matrices M_1 and M_2, we have $Cov(M_1, M_2) = \langle M_1 M_2 \rangle - \langle M_1 \rangle \langle M_2 \rangle = \langle M_2 M_1 \rangle - \langle M_2 \rangle \langle M_1 \rangle = Cov(M_2, M_1)$, so covariance matrices are symmetric around the main diagonal.

We now return to the pathological example from Section 4.10.1. Since the variance of a partitioning is the sum of all entries in each partition's covariance matrix, the sum of the shaded terms below represents the variance of the $k = 2$ partitioning $(\{-XX, -YY, ZZ\}, \{ZI, IZ\})$:

$$\begin{pmatrix} \sigma^2_{-XX} & \sigma_{-XX,-YY} & \sigma_{-XX,ZZ} & \sigma_{-XX,ZI} & \sigma_{-XX,IZ} \\ \sigma_{-YY,-XX} & \sigma^2_{-YY} & \sigma_{-YY,ZZ} & \sigma_{-YY,ZI} & \sigma_{-YY,IZ} \\ \sigma_{ZZ,-XX} & \sigma_{ZZ,-YY} & \sigma^2_{ZZ} & \sigma_{ZZ,ZI} & \sigma_{ZZ,IZ} \\ \sigma_{ZI,-XX} & \sigma_{ZI,-YY} & \sigma_{ZI,ZZ} & \sigma^2_{ZI} & \sigma_{ZI,IZ} \\ \sigma_{IZ,-XX} & \sigma_{IZ,-YY} & \sigma_{IZ,ZZ} & \sigma_{IZ,ZI} & \sigma^2_{IZ} \end{pmatrix}$$

And the sum of the shaded terms below represents the variance of the $k = 3$ partitioning $(\{-XX\}, \{-YY, ZZ\}, \{ZI, IZ\})$:

$$\begin{pmatrix} \sigma^2_{-XX} & \sigma_{-XX,-YY} & \sigma_{-XX,ZZ} & \sigma_{-XX,ZI} & \sigma_{-XX,IZ} \\ \sigma_{-YY,-XX} & \sigma^2_{-YY} & \sigma_{-YY,ZZ} & \sigma_{-YY,ZI} & \sigma_{-YY,IZ} \\ \sigma_{ZZ,-XX} & \sigma_{ZZ,-YY} & \sigma^2_{ZZ} & \sigma_{ZZ,ZI} & \sigma_{ZZ,IZ} \\ \sigma_{ZI,-XX} & \sigma_{ZI,-YY} & \sigma_{ZI,ZZ} & \sigma^2_{ZI} & \sigma_{ZI,IZ} \\ \sigma_{IZ,-XX} & \sigma_{IZ,-YY} & \sigma_{IZ,ZZ} & \sigma_{IZ,ZI} & \sigma^2_{IZ} \end{pmatrix}$$

Therefore, it is favorable (fewer state preparations needed to achieve a target accuracy) to break the $-XX$ term out of the $\{-XX, -YY, ZZ\}$ partition if the condition atop the next page holds. The matrices represent a sum over enclosed terms, and the multiplicative factors of $k = 2$ and $k = 3$ follow from Equation 4.2.

$$2\begin{pmatrix} \sigma^2_{-XX} & \sigma_{-XX,-YY} & \sigma_{-XX,ZZ} & & \\ \sigma_{-YY,-XX} & \sigma^2_{-YY} & \sigma_{-YY,ZZ} & & \\ \sigma_{ZZ,-XX} & \sigma_{ZZ,-YY} & \sigma^2_{ZZ} & & \\ & & & \sigma^2_{ZI} & \sigma_{ZI,IZ} \\ & & & \sigma_{IZ,ZI} & \sigma^2_{IZ} \end{pmatrix} > 3\begin{pmatrix} \sigma^2_{-XX} & & & & \\ & \sigma^2_{-YY} & \sigma_{-YY,ZZ} & & \\ & \sigma_{ZZ,-YY} & \sigma^2_{ZZ} & & \\ & & & \sigma^2_{ZI} & \sigma_{ZI,IZ} \\ & & & \sigma_{IZ,ZI} & \sigma^2_{IZ} \end{pmatrix}$$

or equivalently, if:

$$\begin{pmatrix} & 2\sigma_{-XX,-YY} & 2\sigma_{-XX,ZZ} & & \\ 2\sigma_{-YY,-XX} & & & & \\ 2\sigma_{ZZ,-XX} & & & & \\ & & & & \\ & & & & \end{pmatrix} > \begin{pmatrix} \sigma^2_{-XX} & & & & \\ & \sigma^2_{-YY} & \sigma_{-YY,ZZ} & & \\ & \sigma_{ZZ,-YY} & \sigma^2_{ZZ} & & \\ & & & \sigma^2_{ZI} & \sigma_{ZI,IZ} \\ & & & \sigma_{IZ,ZI} & \sigma^2_{IZ} \end{pmatrix} \tag{4.3}$$

Informally, notice that the left-hand side of Equation 4.3 is a multiple of the sum of the covariances that exist in the expression for $Var(k = 2)$ but not $Var(k = 3)$ (which we will call the "broken terms"), whereas the right-hand side is a multiple of the sum of the variances and covariances that exist in both the $Var(k = 2)$ and $Var(k = 3)$ expressions (the "unbroken terms"). This pattern generalizes such that it is favorable to switch from a partitioning with k partitions to a clique-splitting partitioning with $k' > k$ partitions if:

$$k * (\sum \text{broken terms}) > (k' - k) * (\sum \text{unbroken terms})$$

A similar strategy was described in [90, Section V. A.], for the special case of comparing Naive partitions (with no covariances) with QWC partitions; our work generalizes to the case of comparing two non-Naive partitions where both sides have covariance terms.

4.10.4 Strategies for covariance estimation

As demonstrated in Section 4.10.1, the expected number of state preparations needed to determine $\langle H \rangle$ to an accuracy level ϵ can be calculated if the variances and pairwise covariances of commuting Pauli terms under an ansatz state are known.

In practice, the true theoretical values of these variances cannot be known beforehand, as doing so would require computations involving the exponentially sized ansatz state vector. However, just as we use repeated measurements from partitions of commuting terms to approximate the expected value of their sum, **we can use these same measurements to approximate the covariance matrices of Pauli strings in the same partition**. This estimation of covariance is termed "sample covariance", since its value is calculated via a sample from the theoretical distribution. This key idea of adaptively building a sample covariance matrix, using the measurements we are already making, allows us to adaptively detect and correct for harmful covariance terms.

Note that the theoretical variance of $\langle M \rangle$ is $Var(M) = \langle M^2 \rangle - \langle M \rangle^2$, and is approximated by the sample variance, $\widehat{Var}(M) = \frac{1}{n-1}\sum_{i=1}^{n}(m_i - \overline{m})$, where $\{m_1, ..., m_n\}$ represent the n observed measurements of M, and where $\overline{m} = \frac{1}{n}\sum_{i=1}^{n} m_i$ is the sample mean. Similarly, the theoretical covariance $Cov(M_1, M_2) = \langle M_1 M_2 \rangle - \langle M_1 \rangle \langle M_2 \rangle$ is approximated by the sample covariance $\widehat{Cov}(M_1, M_2) = \frac{1}{n-1}\sum_{i=1}^{n}(m_{1i} - \overline{m_1})(m_{2i} - \overline{m_2})$ where $\{m_{11}, ..., m_{1n}\}$ and $\{m_{21}, ..., m_{2n}\}$ are the n observed measurements of M_1 and M_2 respectively.

Since covariance terms can only be approximated if terms are simultaneously measured, we ideally want to start our measurements in a setting with MIN-COMMUTING-PARTITIONS. Fortunately, this is exactly the optimal starting strategy that we initialize with, as per the argument in Section 4.10.2. Once we collect sufficiently many observations that the sample covariance matrices stabilize, this will enable us to identify opportunities to split partitions in order to lower variances and thus reduce the number of requisite state preparations.

To make this concrete, let us again consider the $k = 2$ partitioning from the previous example, $\{-XX, -YY, ZZ\}, \{ZI, IZ\}$. As we accumulate more observations, we can empirically build up an approximation of each partition's sample covariance matrices, like so:

$$\begin{pmatrix} \hat{\sigma}^2_{-XX} & \hat{\sigma}_{-XX,-YY} & \hat{\sigma}_{-XX,ZZ} & & \\ \hat{\sigma}_{-YY,-XX} & \hat{\sigma}^2_{-YY} & \hat{\sigma}_{-YY,ZZ} & & \\ \hat{\sigma}_{ZZ,-XX} & \hat{\sigma}_{ZZ,-YY} & \hat{\sigma}^2_{ZZ} & & \\ & & & \hat{\sigma}^2_{ZI} & \hat{\sigma}_{ZI,IZ} \\ & & & \hat{\sigma}_{IZ,ZI} & \hat{\sigma}^2_{IZ} \end{pmatrix}$$

Since the sample covariance matrix $\widehat{Var}(k=2)$ contains a superset of the terms needed to calculate $\widehat{Var}(k=3)$, we can use observations from the $k=2$ setting to explore whether further partitions would be beneficial.

Each of the grey lines in Figure 4.19 depicts the value of $\widehat{Var}(k=2) - \widehat{Var}(k=3)$ as it evolves with a set of 100 observed measurements under the $|01\rangle$ state. The plot illustrates that the empirical difference, $\widehat{Var}(k=2) - \widehat{Var}(k=3)$ converges to the true theoretical difference, $Var(k=2) - Var(k=3) = 2$ after around 30 observations. The positive sign of this difference indicates that $Var(k=3) < Var(k=2)$, and therefore the $k=3$ partitioning should be favored due to its lower variance.

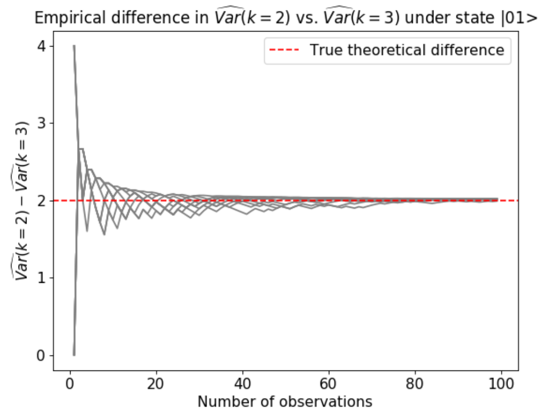


Figure 4.19: Convergence of the empirical difference, $\widehat{Var}(k=2) - \widehat{Var}(k=3)$, to the true difference in variances under $|01\rangle$. Since $Var(k=2) - Var(k=3)$ is positive, this signals that the $k=3$ partitioning will lead to a lower-variance estimator.

When we broaden analysis of the $k = 2$ versus $k = 3$ setting across many different random states, we observe that the state $|01\rangle$ is indeed atypical and pathological, as suggested in Section 4.10.2. Under the vast majority of states, the variance of the $k = 2$ setting is lower than the $k = 3$ setting, as observed by the negative values of $\widehat{Var}(k = 2) - \widehat{Var}(k = 3)$ in Figure 4.20, and therefore the $-XX$ term should not be split into a separate partition.

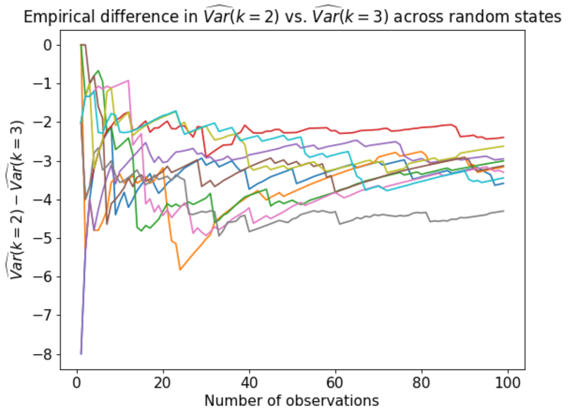


Figure 4.20: The empirical difference in $\widehat{Var}(k = 2) - \widehat{Var}(k = 3)$ across ten Haar-randomly-chosen states. While the convergence value differs across states, it is is negative in all ten cases. This contrasts with the atypical case of convergence to a positive value in the example of Figure 4.19 under state $|01\rangle$.

This discussion naturally leads to the question of how many observations are necessary for the sample covariance matrix to be a good approximation of the true theoretical covariance matrix. To answer this question, we need to formalize a notion of the accuracy of a sample covariance matrix. Several candidate measures may be considered, which we are exploring in ongoing work:

- Enforcing a minimum number of "burn-in" observations. This acts as a proxy of the sample observations being sufficiently representative of the true theoretical distribution.
- Enforcing that the distance between the sample covariance matrix after $n-1$ observations and after n observations be less than a pre-specified threshold. This acts as an alternative proxy of the stability of the observations on which the sample covariance matrix is

based.

- Enforcing that a hypothesis test between the sample variance of the full partitioning and the sample variance of the split-up partitioning returns a p-value below a pre-specified significance level.

The last candidate measure is the most attractive because p-values can be compared across different experimental settings. By contrast, appropriate cutoff values for the first two measures vary with H and $|\psi\rangle$. Formalization of the last measure will require further work to confirm the distribution of the sample variance and covariance terms.

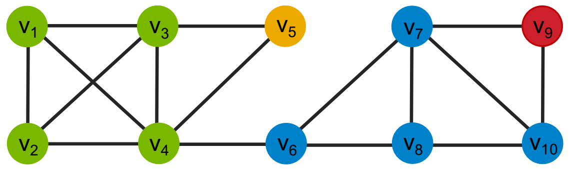
4.11 Conclusion

Our techniques and demonstrations show that simultaneous measurement substantially reduces the cost of Variational Quantum Eigensolver by allowing state preparations to cover several Pauli strings simultaneously. We demonstrate algorithms that achieve up to 30x reductions in the number of requisite state preparations. We also raise practical concerns about these algorithms and identify an alternate strategy that exploits properties of molecular Hamiltonians to achieve an 8x reduction in state preparation cost, with almost no additional pre-computation. Our systems emphasis includes explicit attention to the overhead of simultaneous measurement circuits. Accordingly, we develop a circuit synthesis procedure, which we have implemented and tested in software. We also study the statistics of simultaneous measurement, and ensure that the top-level goal of finding MIN-COMMUTING-PARTITIONs is statistically justified. Our statistical analysis also yields a strategy for detecting and correcting course when simultaneous measurements are harmed by covariance terms. Our theoretical and benchmark/simulation results are accompanied by a proof-of-concept experimental validation on the IBM 20Q quantum computer.

Our ongoing work includes further benchmarking, more theoretical investigation, and the development of a software tool that packages together all of our techniques. We also see promising future work towards further developing molecular-Hamiltonian-aware partitioning strategies, especially since the advantage of the MIN-COMMUTING-PARTITION appears to improve with molecular size. Moreover, other qubit encodings like Bravyi-Kitaev, as well as Hamiltonian reduction techniques such as active space reductions and frozen orbitals should be considered.

4.12 MIN-COMMUTING-PARTITION is NP-Hard

We show that MIN-COMMUTING-PARTITION is NP hard. Given a set of operators $o_1, o_2, \ldots, o_n$, the MIN-COMMUTING-PARTITION problem partitions the operator set into k subsets such that all operators in each subset pairwise commute and k is minimized. The corresponding decision problem is in NP as it is easy to verify pairwise commutativity for each subset of operators. To show NP completeness it remains to show the problem is NP hard. This can be done by reducing from MIN-CLIQUE-COVER. Given a graph $G = (V, E)$ with n vertices that represents an instance of MIN-CLIQUE-COVER, we produce an instance of MIN-COMMUTING-PARTITION consisting of a set of operators $o_1, o_2, ..., o_n$ where each operator o_i has n Paulis, and the j-th Pauli is Z if $j = i$, X if $j > i$ and $(v_i, v_j) \notin E$, and I otherwise. This is illustrated in Figure 4.21. It is easy to see that a commuting subset of operators corresponds to a clique, which concludes the proof. Notice that the commutativity relationships required in this reduction are only Qubit-Wise Commutative, meaning that even the QWC-restricted MIN-COMMUTING-PARTITION problem is NP-Hard.



	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
o_1:	Z	I	I	I	X	X	X	X	X	X
o_2:	I	Z	I	I	X	X	X	X	X	X
o_3:	I	I	Z	I	I	X	X	X	X	X
o_4:	I	I	I	Z	I	I	X	X	X	X
o_5:	I	I	I	I	Z	X	X	X	X	X
o_6:	I	I	I	I	I	Z	I	I	X	X
o_7:	I	I	I	I	I	I	Z	I	X	I
o_8:	I	I	I	I	I	I	I	Z	X	I
o_9:	I	I	I	I	I	I	I	I	Z	I
o_{10}:	I	I	I	I	I	I	I	I	I	Z

Figure 4.21: Instance of MIN-CLIQUE-COVER (top) and MIN-COMMUTING-PARTITION (bottom).

CHAPTER 5
$O(N^3)$ MEASUREMENT COST FOR VARIATIONAL QUANTUM EIGENSOLVER ON MOLECULAR HAMILTONIANS

5.1 Background

Variational Quantum Eigensolver (VQE) [84] is a quantum algorithm that is a leading contender, if not the top contender, for demonstrating a practical quantum advantage on near-term machines. Unlike traditional quantum algorithms, which have extremely high quantum requirements in terms of gate counts and qubit lifetimes, VQE is feasible with modest quantum resources that are already available on current quantum computers. It attains a lower quantum resource cost in part by structuring computation over a large number of subproblems, each of which can be performed on a quantum computer with modest capabilities.

While the low quantum resource requirements per subproblem are appealing, the number of subproblems is an issue for practical application of VQE. Consider molecular ground state estimation, a classically-hard problem that is considered the canonical application of VQE. Within the framework of VQE, molecular energy estimation is performed by applying linearity of expectation to the Hamiltonian H, an observable that captures a molecule's energy configuration. Under the second quantization and expressed in fermionic form, we have [109]:

$$H = \sum_{p=1}^{N}\sum_{q=1}^{N} h_{pq} a_p^\dagger a_q + \sum_{p=1}^{N}\sum_{q=1}^{N}\sum_{r=1}^{N}\sum_{s=1}^{N} h_{pqrs} a_p^\dagger a_q^\dagger a_r a_s \tag{5.1}$$

Applying linearity of expectation, we see that measuring $\langle H\rangle$ reduces to measuring $\langle a_p^\dagger a_q\rangle$ and $\langle a_p^\dagger a_q^\dagger a_r a_s\rangle$. Each of these $O(N^4)$ terms is transformed via fermion-to-qubit encoding into a sum over a constant number of Pauli strings (N-fold tensor product of Pauli matrices). Measurement of each of these resulting $O(N^4)$ Pauli strings constitutes a subproblem. Although the measurement for each subproblem is simple, requiring only single-qubit rotations, the $O(N^4)$ scaling of subproblems poses a practical challenge towards applying VQE to molecules of interest such as caffeine and cholesterol, which appear to require N numbering hundreds of qubits [188].

Recently however, several research groups observed that this $O(N^4)$ scaling may be reducible to $O(N^3)$ [19, 156, 158, 159, 189]. The core principle underlying these papers is that commuting Pauli strings can be measured simultaneously. The $O(N^4) \to O(N^3)$ improvement is conjectured based on extrapolation of results across a range of molecules.

Here, we confirm this observation of linearly-reduced measurement cost for molecular Hamiltonians encoded under Jordan-Wigner—the most widely used encoding [190]. Our general approach is to demonstrate that the molecular Hamiltonians can always be partitioned into pairwise-commuting families where each family contains $O(N)$ terms. Since the terms in each such family can be measured simultaneously, this constitutes our reduction in the measurement cost of VQE from $O(N^4)$ to $O(N^3)$.

In addition to proving the existence of such a partition, we explicitly demonstrate how to construct it. Our construction is efficient, computable in $O(N^5 \log N)$ time. Moreover, the construction is independent of the specific molecular Hamiltonian of interest and instead only depends on N. This means that the partitioning can be pre-computed once for each N. The efficiency of our approach is critical. In contrast, proposals for simultaneous measurement in the recent prior work have involved algorithms with runtimes as high as $O(N^{12})$, which may be slow enough to undermine the advantage of simultaneous measurement.

5.2 Prior Work

The empirical results in [19, 156, 158, 159, 189] all suggest that the advantage due to simultaneous measurement appears to increase for larger molecules. The specificity of this claim varies across the papers—[19] explicitly extrapolates linear scaling for molecular Hamiltonians over a range of encodings, molecules, and active space sizes; [156] formulates it as an explicit conjecture for "almost all" sets of Pauli strings; [158, 159] observes this scaling via least-squares fitting for molecular Hamiltonians under the Jordan-Wigner and Bravyi-Kitaev qubit encodings; and [189] makes note of increasing partition size with increasing N.

Moreover, [19, Section 5.1] provides two encouraging examples of an asymptotic gain from simultaneous measurement for *specific* types of contrived Hamiltonians. First, it is observed that simultaneous measurement can yield an exponential gain: the 2^N Pauli strings with the same underlying measurement basis across all qubits can be simultaneously measured with a single measurement. Second, in the case of measuring all 4^N Pauli strings on N qubits, a square root (2^N) reduction is achievable by Mutually Unbiased Bases.

However, as suggested by [19, Appendix A], an asymptotic gain from simultaneous measurement is not guaranteed. For example, consider the set of $2N$ Pauli strings matching the pattern Z*(X|Y)I*, where * matches 0 or more occurrences and | is a Boolean OR. For example, for $N = 3$, we have $[XII, YII, ZXI, ZYI, ZZX, ZZY]$. It can be shown that none of the pairs in this set commute. Thus, simultaneous measurement offers no advantage for this set of Pauli strings. More generally, we see that simultaneous measurement does not automatically confer any advantage.

During the preparation of this manuscript, we became aware of very recent work by [191] that also proves the $O(N^3)$ measurement cost for molecular Hamiltonians. Their work approaches the problem via Majorana operators, which leads to a proof agnostic of the underlying fermion-to-qubit encoding.

5.3 Commutativity of Index-Disjoint Terms

Our top-level goal is to partition the molecular Hamiltonian into commuting families, such that the number of partitions is minimized. This problem is termed MIN-COMMUTING-PARTITION and is NP-Hard in general [19]. We instead seek to *approximate* a good partitioning. Our approach is to address this problem at the level of the fermionic Hamiltonian in Equation 5.1. By contrast, past work, except for [19, Section 6], has focused on this problem at the qubit Hamiltonian stage, after the fermionic Hamiltonian has been encoded into a summation over Pauli strings.

We focus on the $O(N^4)$ terms with $p \neq q \neq r \neq s$ in the second sum of Equation 5.1, because these terms are asymptotically dominant; the number of other terms is only $O(N^3)$. Without loss of generality, let us suppose that $p > q > r > s$, and likewise $i > j > k > l$. We denote the set of Pauli strings in the Jordan-Wigner encoding of $a_p^\dagger a_q^\dagger a_r a_s$ as $\{a_p^\dagger a_q^\dagger a_r a_s\}_{\text{JW}}$.

Our core observation is that if two $a^\dagger a^\dagger a a$ terms have disjoint indices, then the terms in their qubit encodings commute. In particular:

Theorem 3. *If $\{p, q, r, s\} \cap \{i, j, k, l\} = \emptyset$, then*

$$[\{a_p^\dagger a_q^\dagger a_r a_s\}_{JW}, \{a_i^\dagger a_j^\dagger a_k a_l\}_{JW}] = 0$$

where the commutator is taken to apply between all pairs of elements between the two sets.

Theorem 3 can be verified by inspecting the form of the Pauli string terms in $\{a^\dagger a^\dagger a a\}_{\text{JW}}$. Under the Jordan-Wigner encoding [146], we perform the transformations:

$$a_p \rightarrow \frac{X_p + iY_p}{2} Z_{p-1} ... Z_0, \quad a_p^\dagger \rightarrow \frac{X_p - iY_p}{2} Z_{p-1} ... Z_0$$

Carrying out the transformation for $a_p^\dagger a_q^\dagger a_r a_s$ yields the 16 Pauli strings matching the regular expression:

$$(X_p|Y_p)\overline{Z}_{p:q}(X_q|Y_q)(X_r|Y_r)\overline{Z}_{r:s}(X_s|Y_s)$$

where $\overline{Z}_{p:q}$ denotes Z on each index between p and q, *exclusive* of endpoints. Figure 5.1 shows this pattern as a pictorial representation: the repeating Z's are blue rectangles and the $\{p, q, r, s\}$ indices are the black vertical bars demarcating the blue and white rectangles.

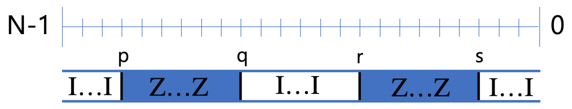


Figure 5.1: Pictorial representation of the Jordan-Wigner encoding of $a_p^\dagger a_q^\dagger a_r a_s$. Repeating Z's span the blue rectangles between p and q and between r and s. The other three ranges have repeating I's. At indices p, q, r, and s, which are denoted by the black vertical bars between the blue and white rectangles, we can have either X or Y. Thus, there are $2^4 = 16$ Pauli strings involved in the Jordan-Wigner encoding.

To evaluate the commutativity between a term in $\{a_p^\dagger a_q^\dagger a_r a_s\}_{\text{JW}}$ and a term in $\{a_i^\dagger a_j^\dagger a_k a_l\}_{\text{JW}}$, we simply need to count the number of indices that anti-commute, as explained in [19, Section 3]. If the number of anti-commuting indices is even, then the two Pauli strings commute. For all indices other than p, q, r, s, i, j, k, l, the Pauli matrices at the indices commute, because $[I, I] = [I, Z] = [Z, I] = [Z, Z] = 0$. On the remaining 8 indices, the commutation depends on whether the $(X|Y)$ is matched to an I (commutes) or Z (anti-commutes). Figure 5.2 depicts this: when one of the black bars $(X|Y)$ is vertically aligned with a blue rectangle (Z), the index does not commute, as marked by the red cross. When the black bar is vertically aligned with a white rectangle (I), the index commutes.

The commutativity between $\{a_p^\dagger a_q^\dagger a_r a_s\}_{\text{JW}}$ and $\{a_i^\dagger a_j^\dagger a_k a_l\}_{\text{JW}}$ terms can be verified by considering all possible interleaved orderings of the 8 indices, subject to the constraint that $p > q > r > s$ and $i > j > k > l$. There are $\binom{8}{4} = 70$ such cases that can be explicitly checked (or 35 cases, accounting for symmetry) to prove Theorem 3. Figure 5.3 demonstrates four representative cases, which provide useful intuition for the general case. In particular, when sliding one of the $\{p, q, r, s\}$ indices while keeping $\{i, j, k, l\}$ fixed, the parity of the

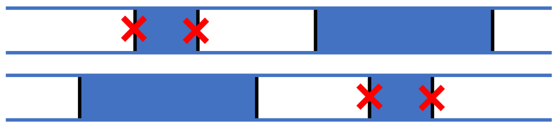


Figure 5.2: Pictorial representation of the commutation on each index between two $\{a^\dagger a^\dagger aa\}_{\text{JW}}$ rectangles. All indices commute except possibly the 8 indices with black bars—these indices anti-commute when the black bar (X or Y) is vertically aligned with a blue rectangle Z. In this example, the there are an even (4) number of anti-commuting terms, so the two patterns commute.

number of anti-commuting indices is invariant. Thus, this parity is always even, and two $\{a_p^\dagger a_q^\dagger a_r a_s\}_{\text{JW}}$ and $\{a_i^\dagger a_j^\dagger a_k a_l\}_{\text{JW}}$ terms with disjoint indices always commute, as claimed in Theorem 3.

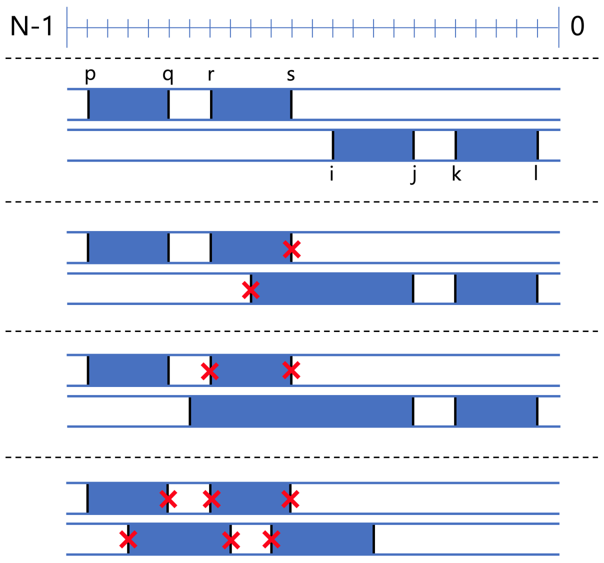


Figure 5.3: Four representative examples illustrating why $\{a_p^\dagger a_q^\dagger a_r a_s\}_{\text{JW}}$ and $\{a_i^\dagger a_j^\dagger a_k a_l\}_{\text{JW}}$ terms always commute (have an even number of anti-commuting indices) when $\{p, q, r, s\} \cap \{i, j, k, l\} = \emptyset$. At the top, no black bars align with blue rectangles, so there are 0 anti-commuting indices. Below, $r > i > s > j$, so there are 2 anti-commuting indices: i and s. Below that, observe that sliding the i endpoint into the interval between q and r does not change the parity of the number of anti-commuting indices. The bottom example shows a case with the maximal number of anti-commuting indices, 6.

5.4 Existence of Linearly-Sized Partitions

Consider the set of Pauli strings contained in

$$\{a_N^\dagger a_{N-1}^\dagger a_{N-2} a_{N-3}\}_{\mathrm{JW}} \cup \{a_8^\dagger a_7^\dagger a_6 a_5\}_{\mathrm{JW}} \cup \ldots \cup \{a_4^\dagger a_3^\dagger a_2 a_1\}_{\mathrm{JW}}$$

for N divisible by 4. There are $16\frac{N}{4} = 4N \in O(N)$ Pauli strings in this set. However, since the indices are disjoint, Pauli strings from each of the $\frac{N}{4}$ subsets can be measured simultaneously by Theorem 3. In particular, the Pauli strings can be partitioned into $16 \in O(1)$ measurement families. In fact, they can even be partitioned into just 2 measurement families by noting that the MIN-COMMUTING-PARTITION within each $\{a_p^\dagger a_q^\dagger a_r a_s\}_{\mathrm{JW}}$ term is 2, as described in [19, Section 6].

A natural question is whether **all** $\binom{N}{4}$ $p > q > r > s$ terms in Equation 5.1 can be partitioned in such a fashion—if so, then this constitutes a partitioning of the $O(N^4)$ terms into $\binom{N}{4}/\frac{N}{4} = \binom{N-1}{3} \in O(N^3)$ commuting families. Intuitively, this is the same problem as trying to schedule a round-robin tournament of N players with 4 players-per-game into $\binom{N-1}{3}$ rounds. We can think of each index as a player, and 4-player games can be scheduled simultaneously if they don't share players. Equivalently, these problems can be bijected to a graph theory problem: does the 4-uniform complete hypergraph on N vertices admit a 1-factorization?

The answer to all of these questions is affirmative, per Baranyai's Theorem [8]. In our case, it means that for N divisible by 4, the $\binom{N}{4} \in O(N^4)$ terms can be partitioned into $\binom{N-1}{3} \in O(N^3)$ sets, such that the $\frac{N}{4}$ terms within each set have disjoint indices. Table I demonstrates such a partitioning for $N = 8$ qubits. Each of the $\binom{8-1}{3} = 35$ rows has two fermionic terms with disjoint indices—thus, their corresponding Jordan-Wigner qubit encodings can be measured simultaneously.

5.5 Construction of Linearly-Sized Partitions

Prior literature refers to Baranyai's original proof as either being non-constructive [192, 193] or providing an exponential-time construction [194] (prior literature varies in what exactly is considered Baranyai's proof). In order for Baranyai's proof to be useful to us, we need a fast polynomial-time algorithm for partitioning the $\binom{N}{4}$ subsets of N into $\binom{N-1}{3}$ groups, each containing $N/4$ disjoint subsets. Fortunately, due to later work by [195], a proof was provided that leads to an efficient construction [196]. The proof is based on maximum flows in network flow graphs.

We refer readers to [197] for a lucid explanation and to [198] for an implementation in code. This implementation was used to generate Table I. The pseudocode is given in Algorithm 4. An outer loop is called N times, and each iteration solves for maximum flow on a network with $O(N^3)$ vertices and $O(N^4)$ directed edges. Since the maximum flow in the proof construction has a value of $O(N^3)$, solving for it with the Ford-Fulkerson algorithm would incur a cost of $O(N^7)$ per loop iteration [199]. However, due to work on flow-rounding [200, 201, 202, 203], this runtime is reduced to $O(N^4 \log N)$. This is because for each flow network, a *fractional* solution is known that can be rounded to an integral solution faster than computing an integral solution from scratch. Thus, the total runtime of the Baranyai constructive proof is $O(N^5 \log N)$.

A useful aspect of the Baranyai-based approach to molecular Hamiltonian partitioning is that it depends only on N and not on the h_{pq} and h_{pqrs} coefficients in Equation 5.1. In this sense, it is pre-computable—for instance, the $N = 8$ partitioning in Table I will apply to all 8-qubit Hamiltonians. By contrast, MIN-COMMUTING-PARTITION techniques in prior work operate on the specific molecular Hamiltonians of interest. Thus, the partitionings are not pre-computable and the classical cost of partitioning must be accounted for in time-to-solution.

Algorithm 4: $O(N^5 \log N)$ Baranyai construction

```
input  : $N$
output : $\binom{N-1}{3}$ sets where each set contains $\frac{N}{4}$ disjoint size-4 subsets, and no term
         is repeated
for $i \in [1, 2, ..., N]$ do
    Create flow network with two layers: $O(N^3)$ partition nodes and $O(N^3)$ subset
      nodes
    Set capacities for $O(N^4)$ edges per [195, 197] construction
    Set fractional maxflow of value $\binom{N-1}{3}$, saturating nodes out of source and into
      destination
    Round fractional maxflow into an integral maxflow
    Update subset nodes based on integral maxflow
end
Return schedule of $\binom{N}{4}$ subsets, based on final flow
```

5.6 Discussion

We have demonstrated that Jordan-Wigner encoded molecular Hamiltonians can be partitioned into $O(N^3)$ commuting families, each containing $O(N)$ Pauli strings. Our proof stems from Baranyai's Theorem, which has a constructive form that efficiently yields partitionings, per Algorithm 4. Since commuting families can be measured simultaneously, this constitutes a reduction in the measurement cost of VQE from $O(N^4)$ naively to $O(N^3)$ with these partitions. The simultaneous measurement circuits are efficient too, requiring only $O(N)$ gates, since the shared eigenbasis of the commuting partitions can be expressed as a tensor product over 4-qubit chunks.

An advantage of our technique is that it only depends on N and is pre-computable for all N-qubit molecular Hamiltonians. Further optimizations may be possible by analyzing h_{pqrs} coefficients in Equation 5.1. For example, for molecular Hamiltonians, we expect the $h_{pqrs} = h_{srpq}$ symmetry [172], which reduces the number of relevant Pauli strings in each $\{a_p^\dagger a_q^\dagger a_r a_s\}_{\text{JW}}$ set from 16 to 8.

Recent work [190] has gone deeper in this direction, by factoring molecular Hamiltonians into a form that empirically seems to have $O(N)$ partitions. Moreover, the simultaneous measurements only appear to require $O(N^2)$ gates, even with linear qubit connectivity. It would be informative to benchmark this recent work against our strategy, which produces $O(N^3)$ partitions but requires only $O(N)$ gates under full connectivity.

Beyond VQE, our technique may be useful in other quantum computational chemistry applications. For example, the simulation of Hamiltonian *dynamics* could be improved by partitioning into commuting families. Naively, Hamiltonian evolution is performed by Trotterization that requires fine time slicing to account for non-commuting terms [32, Section 4.7]. However, by ordering Pauli strings in a Hamiltonian such that large commuting sets are consecutive, the Trotterization cost could be diminished. This approach seems promising since our work proves an asymptotic gain for partitioning. Moreover, simultaneous measurement circuits would not be needed, so this re-ordering of a Trotterization would have essentially no quantum cost.

$a_7^\dagger a_5^\dagger a_3 a_0$	$a_6^\dagger a_4^\dagger a_2 a_1$
$a_6^\dagger a_5^\dagger a_3 a_0$	$a_7^\dagger a_4^\dagger a_2 a_1$
$a_7^\dagger a_6^\dagger a_3 a_0$	$a_5^\dagger a_4^\dagger a_2 a_1$
$a_7^\dagger a_4^\dagger a_3 a_0$	$a_6^\dagger a_5^\dagger a_2 a_1$
$a_7^\dagger a_5^\dagger a_4 a_0$	$a_6^\dagger a_3^\dagger a_2 a_1$
$a_6^\dagger a_4^\dagger a_3 a_0$	$a_7^\dagger a_5^\dagger a_2 a_1$
$a_6^\dagger a_5^\dagger a_4 a_0$	$a_7^\dagger a_3^\dagger a_2 a_1$
$a_7^\dagger a_6^\dagger a_4 a_0$	$a_5^\dagger a_3^\dagger a_2 a_1$
$a_5^\dagger a_4^\dagger a_3 a_0$	$a_7^\dagger a_6^\dagger a_2 a_1$
$a_7^\dagger a_6^\dagger a_5 a_0$	$a_4^\dagger a_3^\dagger a_2 a_1$
$a_7^\dagger a_5^\dagger a_1 a_0$	$a_6^\dagger a_4^\dagger a_3 a_2$
$a_7^\dagger a_5^\dagger a_2 a_0$	$a_6^\dagger a_4^\dagger a_3 a_1$
$a_6^\dagger a_5^\dagger a_1 a_0$	$a_7^\dagger a_4^\dagger a_3 a_2$
$a_6^\dagger a_5^\dagger a_2 a_0$	$a_7^\dagger a_4^\dagger a_3 a_1$
$a_7^\dagger a_6^\dagger a_1 a_0$	$a_5^\dagger a_4^\dagger a_3 a_2$
$a_7^\dagger a_4^\dagger a_1 a_0$	$a_6^\dagger a_5^\dagger a_3 a_2$
$a_7^\dagger a_6^\dagger a_2 a_0$	$a_5^\dagger a_4^\dagger a_3 a_1$
$a_7^\dagger a_3^\dagger a_1 a_0$	$a_6^\dagger a_5^\dagger a_4 a_2$
$a_7^\dagger a_4^\dagger a_2 a_0$	$a_6^\dagger a_5^\dagger a_3 a_1$
$a_6^\dagger a_4^\dagger a_1 a_0$	$a_7^\dagger a_5^\dagger a_3 a_2$
$a_6^\dagger a_3^\dagger a_1 a_0$	$a_7^\dagger a_5^\dagger a_4 a_2$
$a_7^\dagger a_3^\dagger a_2 a_0$	$a_6^\dagger a_5^\dagger a_4 a_1$
$a_5^\dagger a_3^\dagger a_1 a_0$	$a_7^\dagger a_6^\dagger a_4 a_2$
$a_6^\dagger a_4^\dagger a_2 a_0$	$a_7^\dagger a_5^\dagger a_3 a_1$
$a_5^\dagger a_4^\dagger a_1 a_0$	$a_7^\dagger a_6^\dagger a_3 a_2$
$a_4^\dagger a_3^\dagger a_1 a_0$	$a_7^\dagger a_6^\dagger a_5 a_2$
$a_6^\dagger a_3^\dagger a_2 a_0$	$a_7^\dagger a_5^\dagger a_4 a_1$
$a_7^\dagger a_2^\dagger a_1 a_0$	$a_6^\dagger a_5^\dagger a_4 a_3$
$a_5^\dagger a_3^\dagger a_2 a_0$	$a_7^\dagger a_6^\dagger a_4 a_1$
$a_6^\dagger a_2^\dagger a_1 a_0$	$a_7^\dagger a_5^\dagger a_4 a_3$
$a_5^\dagger a_2^\dagger a_1 a_0$	$a_7^\dagger a_6^\dagger a_4 a_3$
$a_5^\dagger a_4^\dagger a_2 a_0$	$a_7^\dagger a_6^\dagger a_3 a_1$
$a_4^\dagger a_2^\dagger a_1 a_0$	$a_7^\dagger a_6^\dagger a_5 a_3$
$a_4^\dagger a_3^\dagger a_2 a_0$	$a_7^\dagger a_6^\dagger a_5 a_1$
$a_3^\dagger a_2^\dagger a_1 a_0$	$a_7^\dagger a_6^\dagger a_5 a_4$

Table 5.1: Partitioning of $\binom{N=8}{4} = 70$ $a_p^\dagger a_q^\dagger a_r a_s$ terms into $\binom{N-1=7}{3} = 35$ subsets, with disjoint indices between the two terms in each subset. Such a partitioning is guaranteed to exist for all N divisible by 4, per Baranyai's Theorem [8].

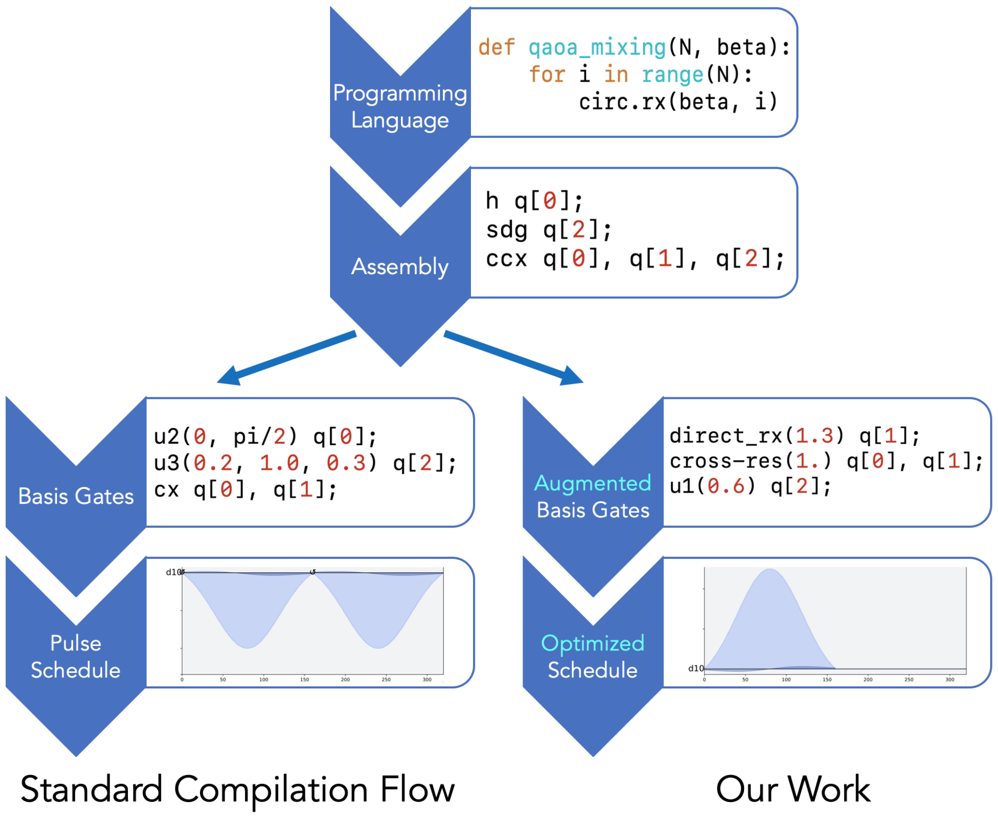
CHAPTER 6
OPTIMIZED QUANTUM COMPILATION FOR NEAR-TERM ALGORITHMS WITH OPENPULSE

6.1 Introduction

The present era of quantum computing is characterized by the emergence of quantum computers with dozens of qubits, as well as new algorithms that have innate noise resilience and modest qubit requirements. There are promising indications that near-term devices could be used to accelerate or outright-enable solutions to problems in domains ranging from molecular chemistry [109] to combinatorial optimization [85] to adversarial machine learning [204]. To realize these practical applications on noisy hardware, it is critical to optimize across the full stack, from algorithm to device.

Standard quantum compilers operate at the level of gates. However, the lowest-level of quantum control is through analog pulses. Pulse optimization has shown promise in previous quantum optimal control (QOC) work [205, 206], but we found that noisy experimental systems are not ready for compilation via QOC approaches. This is because QOC requires an extremely accurate model of the machine, i.e. its Hamiltonian. Hamiltonians are difficult to measure experimentally and moreover, they drift significantly between daily recalibrations. Experimental QOC papers incur significant pre-execution calibration overhead to address this issue. By contrast, we propose a technique that is bootstrapped purely from daily calibrations that are already performed for the standard set of basis gates. The resulting pulses form our **augmented basis gate set**. These pulses are extremely simple, which reduces control error and also preserves intuition about underlying operations, unlike QOC. This technique leads to optimized programs, with mean 1.6x error reduction and 2x speedup for near-term algorithms.

We emphasize the generality of our approach and our compiler, which can target any



```
def qaoa_mixing(N, beta):
    for i in range(N):
        circ.rx(beta, i)
```

```
h q[0];
sdg q[2];
ccx q[0], q[1], q[2];
```

```
u2(0, pi/2) q[0];
u3(0.2, 1.0, 0.3) q[2];
cx q[0], q[1];
```

```
direct_rx(1.3) q[1];
cross-res(1.) q[0], q[1];
u1(0.6) q[2];
```

Figure 6.1: Like classical programs, quantum programs undergo a compilation process from high-level programming language to assembly. However, unlike the classical setting, quantum hardware is controlled via analog pulses. In our work, we optimize the underlying pulse schedule by augmenting the set *basis gates* to match hardware. Our compiler automatically optimizes user code, which therefore remains hardware-agnostic.

underlying quantum hardware. We demonstrate our results via OpenPulse [100, 207], an interface for pulse-level control. In particular, our work is the first experimental demonstration of OpenPulse for optimized compilation of quantum programs (one prior paper used OpenPulse for noise extrapolation [208]). We executed pulse schedules on IBM's 20-qubit Almaden quantum computer, accessible through the cloud via the IBM Q Experience [209]. Our experience-building spanned over 11.4 million experimental shots, 4 million of which are explicitly presented here as concrete research outcomes. Our results indicate that pulse-level control significantly extends the computational capacity of quantum computers. Our techniques are realizable immediately on existing OpenPulse-compatible devices. To this end, all of our code and notebooks are available on Github [210].

We begin with background on quantum computing in Section 6.2. Next, Section 6.3 presents an overview of standard quantum compilers and our compiler design (depicted in Figure 6.1). Sections 6.4– 6.7 describe four key optimizations in our compiler, all of which are enabled by pulse-level control:

1. **Direct Rotations** (Section 6.4). Access to pulse-level control allows us to implement any single-qubit operation *directly* with high fidelity, circumventing inefficiencies from standard compilation.

2. **Cross-Gate Pulse Cancellation** (Section 6.5). Although gates have the illusion of atomicity, the true atomic units are pulses. Our compiler creates new cancellation optimizations that are otherwise invisible.

3. **Two-Qubit Operation Decompositions** (Section 6.6). We recompile important near-term algorithm primitives for two-qubit operations directly down to the two-qubit interactions that hardware actually implements.

4. **Qudit Operations** (Section 6.7). Quantum systems have infinite energy levels. Pulse control enables d-level *qudit* operations, beyond the 2-level qubit subspace.

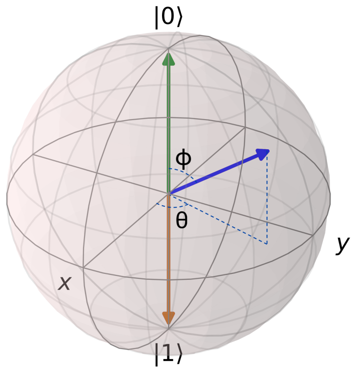


Figure 6.2: The points on the Bloch sphere correspond one-to-one with possible qubit states. The green and brown states correspond to $|0\rangle$ and $|1\rangle$ respectively. The blue state is in a superposition described by latitude and longitude angles.

Section 6.8 presents results from application of these techniques to full algorithms. We conclude in Section 6.9.

6.2 Background

We assume some familiarity with the fundamentals of quantum computing. Here we provide a brief review and expand on elements relevant to our work.

6.2.1 The Qubit

The core unit involved in quantum computation is the qubit (quantum bit). Unlike a classical bit which is either 0 or 1, a qubit can occupy any *superposition* between the two states, which are now denoted $|0\rangle$ and $|1\rangle$. The Bloch sphere, depicted in Figure 6.2, is a useful visual representation of the possible states of a qubit. The North Pole is the $|0\rangle$ state, and the South Pole is the $|1\rangle$ state. The state of a qubit can be parametrized by two angles, latitude and longitude. Upon measurement, a qubit *collapses* to either the $|0\rangle$ or $|1\rangle$ state, with probabilities dependent only on the latitude.

6.2.2 Quantum Gates

The set of valid single-qubit gates correspond to rotations around the Bloch sphere. Arbitrary such rotations are typically decomposed into $R_x(\theta)$ and $R_z(\theta)$ rotations around the X and Z axes respectively, which are universal for single-qubit rotations [211]. A prominent single-qubit gate is the $X = R_x(180^\circ)$ gate, which in Figure 6.2 would rotate the green $|0\rangle$ state 180° around the X-axis to the $|1\rangle$ and vice versa. Thus, the X operation implements the NOT gate.

The set of possible multiple-qubit operations is much richer than the set of single-qubit gates, and lacks a clear visualization on the Bloch sphere. Remarkably however, any multiple-qubit operation can be decomposed into single-qubit rotations + an *entangling* gate such as CNOT [211]. The CNOT gate acts on a control and target qubit, and it applies X to the target iff the control is $|1\rangle$. In part because the CNOT gate is easy to understand, most quantum programs are expressed in terms of it. By implementing a small set of gates: single qubit rotations + CNOT, a quantum computer is universal. Accordingly, quantum computers are generally designed with this interface in mind. However, the lowest level of hardware control is performed by microwave pulses. Foreshadowing the main message of our work: this pulse-backed layer actually provides a richer and "overcomplete" set of gates that outperforms the standard interface for quantum programs.

6.2.3 Gate Calibration

To implement this standard interface of universal gates, quantum computers are routinely calibrated to account for continuous drift in the experimental setting [212, 213]. As a concrete example, for superconducting devices, an $R_x(90^\circ)$ gate is calibrated by performing a Rabi experiment [214, 215, 216] that determines the necessary underlying pulses. An additional DRAG [217, 218, 219] calibration fine-tunes the $R_x(90^\circ)$ gate by cancelling out stray components. Calibrations in a similar spirit are also performed for the two-qubit gate(s).

An interesting feature of the two-qubit gate calibrations is that they have the side effect of also calibrating $R_x(180°)$ pulses on each qubit. We exploit this free calibration in Section 6.4. Typically, $R_Z(\theta)$ rotation gates do not require calibration because they are implemented in software, as described in Section 6.4.

6.2.4 Experimental Setup

Our experiments were performed on IBM's Almaden, a 20 qubit device [220]. Almaden is the first cloud-accessible OpenPulse device. It comprises 20 transmon qubits, with mean T_1 and T_2 coherence lifetimes of 94 and 88 μs respectively. The mean single-qubit and two-qubit (CNOT) error rates are 0.14% and 1.78%. The mean measurement (readout) error was 3.8%, though we used measurement error mitigation [221, 222] to correct for biased measurement errors.

As of December 2019, IBM's publicly cloud-accessible OpenPulse device is the new Armonk device [223], which we used for the most recent results in Figure 6.13. For both Armonk and Almaden (and for IBM's devices in general), the calibrations described above are performed every 24 hours. Our experiments ran around-the-clock via a cloud job queuing system, with varying elapsed time to the prior calibration.

6.3 Compiler Flow

As depicted in Figure 6.1, quantum compilation proceeds through four stages, from high-level to low-level: programming language, assembly, basis gates, and pulse schedule. Also shown is our alternative flow, which creates an augmented set of basis gates and a more optimized pulse schedule. Table 6.1 presents a summary of these four stages. We now discuss existing implementations of these stages, why we should augment the standard set of basis gates, our new compiler framework, and the tradeoffs we considered when we designed the framework.

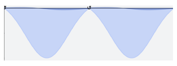
Stage	Notes	Example
Programming Language	High-level; hardware-unaware; sophisticated control flow	`qft(qc)`
Assembly	Usually 1- or 2- qubit arity gates; minimal control flow	`h q[0]`
Basis Gates	Like an HDL; hardware-aware gate set	$\texttt{u}_1\texttt{(3) q[0]}$
Pulse Schedule	Analog waves across channels; ultimate "at the metal" control	

Table 6.1: Summary of the four stages of a quantum compiler.

6.3.1 *Standard Flow*

Programming Language

Quantum PLs are designed to be user-friendly, with sophisticated control flow, debugging tools, and strong abstraction barriers between target operations and underlying quantum hardware. The most successful languages have been implemented as Python packages, such as IBM's Qiskit [224], Google's Cirq [31], and Rigetti's PyQuil [65]. Others are written as entirely new languages, such as Scaffold [225, 88] which is based on LLVM infrastructure; Quipper [103] which is a functional language embedded in Haskell; and Q# [102] which is Microsoft's quantum domain specific language.

Assembly

Quantum assembly languages are closer to hardware, but still aim to be device-agnostic. Generally, the assembly instructions only allow 1- or 2- qubit arity, since hardware primitives act on only 1 or 2 qubits at a time[1]. Quantum assembly is essentially equivalent to the quantum circuit representation of quantum programs. Prominent examples include OpenQASM [100],

1. The notable exception is trapped ion quantum computers, which support global entangling operations that simultaneously act on N qubits [226, 227].

Rigetti's pyQuil [65], and TUDelft's cQASM [228].

Basis Gates

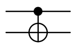
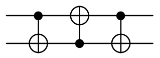
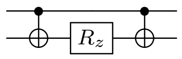
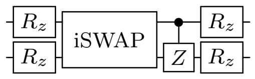
Basis gates are similar to assembly, but re-expressed in terms of the gate set that hardware implements. For example, while the well-known Controlled-Z instruction is valid in assembly code, it would be re-written in basis gates as a sequence of H and $CNOT$ gates—which hardware natively implements. The distinction between assembly and basis gates is primarily a conceptual one; in Qiskit, Cirq, and PyQuil, the basis gate and assembly layers are expressed in the same software framework. In some other domains, for example the Blackbird language [229] for continuous-variable quantum computing, the assembly already resembles a hardware-aware basis gate layer. Regardless of the relationship between between assembly and basis gates, our core observation in this chapter is that existing implementations of basis gate sets are too far from pulse-level hardware primitives. We will expand on this observation for the rest of the chapter.

Pulse Schedule

The ultimate lowest-level control of a quantum computer is a schedule of complex-valued analog pulses, across multiple input channels. The image in Table 6.1 shows a sample pulse schedule on a single channel. The input channels are controlled by an Arbitrary Waveform Generator (AWG) which outputs a continuous value on each channel at every dt. Modern AWGs, such as the one in our experimental realization using IBM's Almaden system, achieve 4.5 Gigasamples per second, i.e. a new complex number every 0.22 ns.

The pulse schedule on drive channel j is referred to as $d_j(t)$ and is complex-norm constrained by $|d_j(t)| \leq 1$. However, qubits are not directly acted on by $d_j(t)$ or $\mathrm{Re}[d_j(t)]$. Instead, the $d_j(t)$ signal is mixed with a *local oscillator* of frequency f_j, leading to a final

Table 6.2: Costs of various two qubit operations, by Native gate. Cost reductions at the right indicate optimization opportunities. One $\sqrt{\text{iSWAP}}$ is treated as 0.5 cost, while iSWAP has 1.0 cost.

		Decomposition Cost by Native Gate						
		“Textbook”	Discrete Gates				Half	**Parametrized**
Operation	Standard Circuit Rep.	CNOT	CR(90°)	iSWAP	bSWAP	MAP	$\sqrt{\text{iSWAP}}$	**CR(θ)**
CNOT		1	1	2	2	1	1	**1**
SWAP		3	3	3	3	3	1.5	**3**
ZZ Interaction		2	2	2	2	2	1	**1**
Fermionic Simulation		3	3	3	3	3	1.5	**3**

signal

$$D_j(t) = \text{Re}[d_j(t)e^{if_jt}] \tag{6.1}$$

This equation will be relevant when we demonstrate qudit operations in Section 6.7.

The translations from basis gates to pulse schedules are known analytically. For example, in superconducting quantum hardware, R_z basis gates are implemented in software with zero-duration and perfect-accuracy via the virtual-Z-Gate translation [93, 230]. The X basis gates is transformed into almost-Gaussian “DRAG” pulses [217, 218, 219]. In the OpenPulse interface, these translations are stored in the `cmd_def` object, and reported by the hardware.

6.3.2 Motivation for Different Basis Gates

At a high level, our core observation is that existing basis gates sets are too far from actual hardware primitives at the pulse-level. This leads to missed opportunities for optimization. Sections 6.4–6.7 will present optimizations resulting from specific gaps between basis gates and pulse-level hardware primitives. Table 6.2 introduces one such gap that we expand upon in Section 6.6. Each row in the table is a two-qubit operation. The columns express the cost[2] of performing the target operation using the given native gate. We computed these costs using Qiskit's `TwoQubitBasisDecomposer` tool, which uses the KAK decomposition [231] described further in [232].

The CNOT column indicates the number of CNOT gates needed to implement the target operation. CNOT is the default "textbook" two-qubit gate, so algorithms are usually written in terms of CNOT. The next group of four columns, Discrete Gates, captures basis gates from

- Fixed-frequency superconducting qubits: 90° Cross-Resonance [233, 234, 235], bSWAP [236], and MAP [237].
- Frequency-tunable superconducting qubits: iSWAP [238] and also CZ [239, 240] which is omitted because it is equivalent to CNOT.
- Quantum dot spin qubits: iSWAP [241]
- Nuclear spin qubits: iSWAP [242]

All four of these columns have identical costs to the CNOT column. As a result of this parity, the prevailing sentiment in current quantum compilation software is that these basis gates are equivalent. Moreover, since quantum algorithms are usually written in terms of CNOTs, there is not an obvious reason to deviate from these basis gates.

The two rightmost columns, challenge this sentiment. The $\sqrt{\text{iSWAP}}$ reflects the fact that quantum hardware allows one to perform "half" of an iSWAP by damping the pulse shape of a standard iSWAP gate. This Half-gate leads to significant improvements over full iSWAPs–each row's cost is halved. The CNOT decomposition and SWAP decomposition are known [243, 244], but to the best of our knowledge, the ZZ Interaction (ubiquitous operation for quantum chemistry and optimization algorithms) and Fermionic Simulation (ubiquitous for quantum chemistry) decompositions are not previously known. They will have immediate

2. Cost here means the number of two-qubit gates needed, since they dominate both error and duration.

applications on hardware that supports $\sqrt{\text{iSWAP}}$ such as frequency-tunable superconducting qubits, quantum dot spin qubits, and nuclear spin qubits.

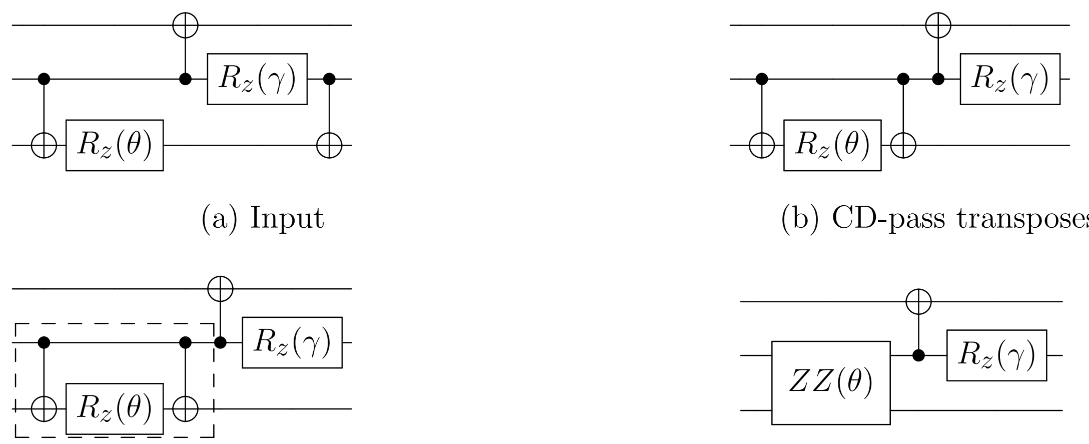
The rightmost column, bolded because it was our experimental target, reflects the fact that fixed-frequency superconducting qubits support parametrized Cross-Resonance(θ) via pulse stretching. Since the native gate is parametrized, we used a different approach to compute the decomposition costs in its column. Specifically, we used the COBYLA constrained optimizer [245] in Scipy [246], with the constraint of finding a 99.9+% fidelity decomposition. Subject to this constraint, our decomposer minimizes the cost of the CR(θ) gates needed to perform the target operation. Observe that ZZ Interaction is 2x cheaper with a Parametrized CR(θ) gate than with the standard CR(90°) gate. The ZZ Interaction is in fact the most common two qubit operation in near-term algorithms. This optimization is expanded upon in Section 6.6.

Our method is extensible to other systems including trapped ions. Some of the trapped ion decompositions have already been studied in recent publications [151, 247, 91].

6.3.3 Design of Our Compiler

Our compiler is implemented as a fork of Qiskit. While Qiskit has traditionally been used in conjunction with IBM superconducting quantum computers, it is a generic framework that supports any underlying quantum hardware. For example, trapped ion quantum computer vendors have recently integrated with Qiskit [248], and OpenPulse support was recently added [249] to the XACC infrastructure for quantum-classical computing [250]. Thus our framework is general, though we performed our experimental realizations on IBM hardware, which is the first to implement OpenPulse.

Our compiler maintains the overall structure of Qiskit, which is already designed with extensibility in mind. As discussed previously, we augment the set of basis gates to better match pulse-level primitives. To support this augmented basis gate set, we re-write the decomposition rules from assembly instructions to basis gates and add new translations



(a) Input

(b) CD-pass transposes gates.

(c) ABGD-pass matches template.

(d) Final

Figure 6.3: Depiction of our compiler passes for commutativity detection (CD) and augmented basis gate detection (ABGD).

(to `cmd_def`) that convert augmented basis gates to pulse schedules. We expand on the augmented basis gates in Sections 6.4–6.7.

To take advantage of the augmented basis gates, we added Qiskit transpiler passes, which convert input quantum assembly into optimized quantum assembly in the spirit of LLVM Transform passes. Our transpiler passes **automatically** optimize user code by using the augmented basis gates. One transpiler pass traverses a DAG-representation of the quantum assembly and pattern matches for templates that represent sequences of gates (such as the ZZ Interaction) that reduce to an augmented basis gate. We also include a commutativity detection transpiler pass that performs this pattern matching even when obfuscated by false dependencies in intermediate gates; this pass is inspired by techniques described in [96]. Figure 6.3 shows an example of these two passes. Through these two passes, we maintain the "write-once target-all" behavior of user-written code, which can remain hardware agnostic.

6.3.4 Compiler Design Tradeoffs

Another compiler design we considered is Quantum Optimal Control [251, 252], which translates directly from the programming language (specifically from the quantum circuit's overall unitary matrix) down to highly optimized pulses. QOC has been explored extensively in physics communities and more recently from an architectural perspective [96, 18].

QOC is indeed a promising path for future machines, and in fact our original aim was to perform pulse-shaping via optimal control. However, our experience revealed experimental roadblocks. In particular, QOC requires a perfect characterization of the quantum computer's underlying physics, i.e. the device Hamiltonian. Pulses designed from an inaccurate Hamiltonian accumulate substantial error. Moreover, to be experimentally realistic, QOC-generated pulses must be constrained to have bounded amplitudes and smooth derivatives. These constraints diminish both the potential advantage of QOC and the reliable convergence of QOC algorithms [97, 18]. In addition, optimizing the pulse shape requires evaluation of partial derivatives of a fidelity metric—a task that is easy analytically or in simulation, but extremely difficult with noisy experimental measurements.

Our experience is mirrored by other work on QOC—the vast majority of prior work has been performed via simulation. The few experimental realizations of QOC generally focus on state preparation (easier than unitary synthesis), e.g. [205, 206]. Moreover, these experiments impose significant Hamiltonian tomography or calibration overhead, for example staggered field calibration [206]. Experimental realizations on superconducting qubits, whose Hamiltonians drift over time [212, 213], are even more rare. In fact, the state-of-art for pulse shaping on superconducting qubits has eschewed standard QOC entirely [253], focusing instead on a closed-loop feedback for tuning pulses. We refer to [253] for further details on the experimental barriers (and opportunities) to QOC, particularly in superconducting qubits. We also note that recent progress in robust control [254] is promising and could justify QOC-based approaches in future work. The work in [254] is already compatible with OpenPulse.

Our approach to pulse-shaping arose from these limitations. In particular, our techniques are bootstrapped from the standard basis gate calibrations, which are already performed daily.

By decomposing and then re-scaling the pre-calibrated pulses, we generate an augmented basis gate set, without ever requiring the device Hamiltonian. We emphasize that our technique can be applied on current cloud-accessible quantum devices, as documented in our Github repository [210]. Moreover, while QOC generally leads to convoluted pulses, our pulses are very simple. This simplicity minimizes the possibility of control errors and also leads to greater interpretability.

6.4 Optimization 1: Direct Rotations

We now present the first of our four optimizations enabled by pulse control. The gist of this optimization is that pulse-level control enables us to perform single-qubit gates (qubit state rotations on the Bloch sphere) via a *direct* trajectory, saving time and potentially reducing errors.

It can be shown that any arbitrary single-qubit gate, termed U_3 in Qiskit, can be implemented by tuning up a single pulse that rotates the qubit state by 90 degrees around the X axis (the $R_x(90°)$ pulse). This is doable due to the following identity, and due to the fact that rotations about the Z axis can be implemented in software at no cost (implemented by a compiler transformation on all future gates involving the target qubit) [93].

$$U_3(\theta, \phi, \lambda) = R_z(\phi + 90°)R_x(90°)R_z(\theta + 180°)R_x(90°)R_z(\lambda) \tag{6.2}$$

The above is extremely attractive from a hardware calibration perspective, since it suggests that fine tuning one pulse is enough to achieve high-fidelity single-qubit gates. In fact, this is how these gates are implemented on IBM quantum computers. We now present experimental evidence that access to one more calibrated gate, as well as pulse control, gives the compiler the ability to optimize single-qubit gates further.

6.4.1 Direct X gates

We first consider the simple X operation, which acts as a NOT by flipping $|0\rangle$ and $|1\rangle$ quantum states. X gates are ubiquitous in algorithms. Our approach relies on access to the $X = R_x(180^\circ)$ rotation, which is already pre-calibrated, as discussed in Section 6.2.3. In our experiments we had access to such a pulse, but one could also be calibrated by the user through OpenPulse. We emphasize that this extra pulse is not strictly necessary for universal computation. However, we use it to demonstrate the power of an *overcomplete* basis for optimizations.

Qiskit's standard compilation flow decomposes an X operation into a U_3 instruction per equation 6.2. At the pulse level, the U_3 instruction is implemented by two consecutive $R_x(90^\circ)$ pulses. Together these complete an X gate (i.e. 180° rotation).

However, the indirection of implementing X with two $R_x(90^\circ)$ pulses becomes unnecessary in the presence of a pre-calibrated $R_x(180^\circ)$ gate. The procedure for calibrating such a gate is very similar to the $R_x(90^\circ)$, and its direct calibration has benefits beyond our discussion here [214, 215, 216, 255]. On IBM hardware enabled with OpenPulse, this pulse is readily available in the backend pulse library.

In our compiler, we exploit this simple observation by augmenting the basis gates with a `DirectX` gate, which is linked to the $R_x(180^\circ)$ pulse that is already calibrated on the quantum computer. This gate is twice as fast as Qiskit's standard X gate, and has 2x lower error, as measured through quantum state tomography experiments.

Figure 6.4 depicts a comparison of pulse schedules used to achieve the X gate in 71.1 ns in the standard framework vs. 35.6 ns in our optimization. It also illustrates why these two pulse schedules are logically equivalent: they have the same (absolute) area-under-curve. To a first approximation—which we will refine below—this area determines how much rotation is applied.

We next consider more sophisticated direct rotation gates, for general angles.

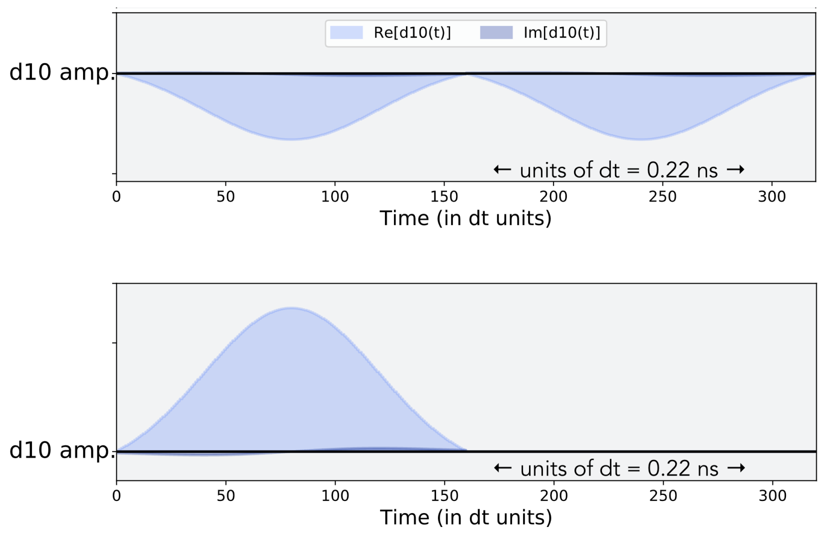


Figure 6.4: Pulse schedules for the X gate via standard compilation (top) versus via direct compilation via our approach (bottom). Time is in units of dt = 0.22 ns. Thus, the `DirectX` gate takes 35.6 ns, twice as fast as the 71.1 ns standard X gate.

6.4.2 Direct partial rotation about the X axis

Since OpenPulse gives us access to arbitrary pulse envelopes, it is natural to ask whether "partial" rotations about the X axis ($R_x(\theta)$ gates) can be realized more efficiently without invoking two discrete $R_x(90°)$ pulses (as done by the standard Qiskit decomposition in Equation 6.2). Our compiler does this by downscaling the amplitude of the pre-calibrated $R_x(180°)$ pulse by $\frac{\theta}{180°}$ to achieve the $R_x(\theta)$ rotation. We represent this as the `DirectRx`(θ) augmented basis gate in our compiler. Since we rely on the pre-calibrated $R_x(180°)$ this technique imposes no calibration overhead.

The results of our experiments with the new `DirectRx`(θ) are summarized in Figure 6.5. Bypassing the gate abstraction, our technique speeds up all R_x rotations by 2x and has 16% lower error on average. We discuss the source of the error reduction in Section 6.8.3.

In the next subsection, we will note how `DirectRx`(θ) generalizes to arbitrary-axis rotations for free.

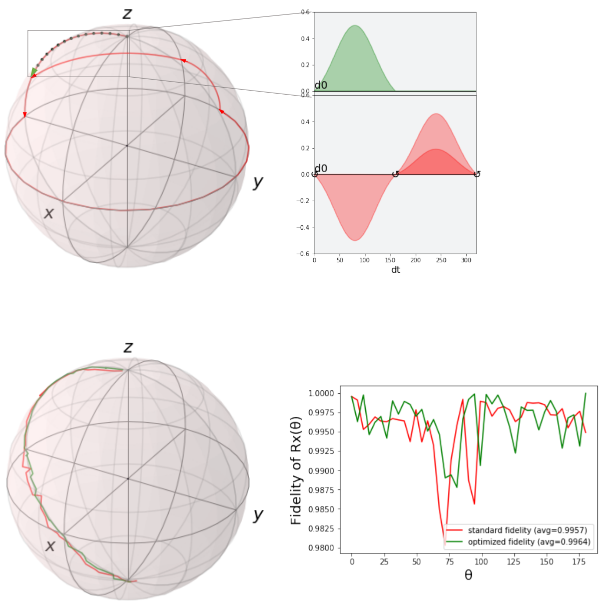


Figure 6.5: Illustration of gate-level vs. pulse-level rotation about the X axis. (top) Trajectory of an $R_x(67°)$ rotation, and the pulses that implement them. Standard gate-based compilation (red) includes two applications of the pre-calibrated $R_x(90°)$ pulse (interleaved with R_z (frame changes) which are zero-cost and in software). Optimized pulse-based compilation takes the shortest path from origin to destination, with only one scaled pulse. (bottom) Fidelity of $R_x(\theta)$ rotations. Each data point is obtained using quantum state tomography experiments to rotate around the X axis by θ. Standard gate-compiled rotations (red) show more jitter from ideal, and 16% higher error on average, compared to optimized pulse-compiled rotations (green).

6.4.3 Optimizing generic rotations

Equipped with an augmented gate set that implements arbitrary X axis rotations at reduced cost, we now show that all single-qubit gates can be achieved with one pulse. Recall that in standard Qiskit compilation, general single qubit gates are implemented via two $R_x(90^\circ)$ pulses and three no-cost R_z frame changes. However, we can write the same gate as [211]:

$$U_3(\theta, \phi, \lambda) = R_z(\phi + 180^\circ) R_x(\theta) R_z(\lambda - 180^\circ) \tag{6.3}$$

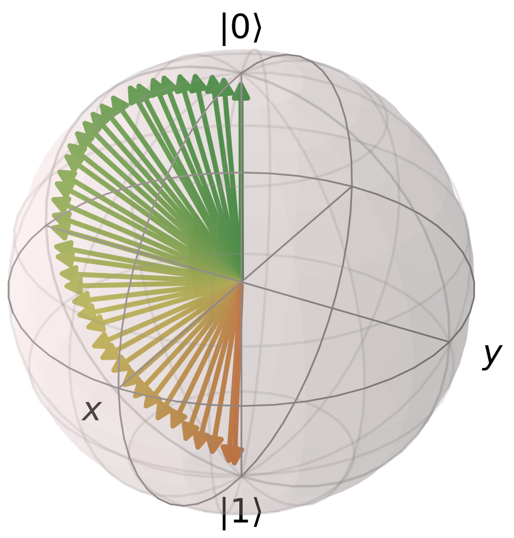
Recall that R_z rotations are implemented by frame changes with perfect fidelity and 0 duration. Thus, this implies that any single-qubit gate can be performed using direct $R_x(\theta)$ rotations, sandwiched by free R_z gates.

6.4.4 Compiler implications

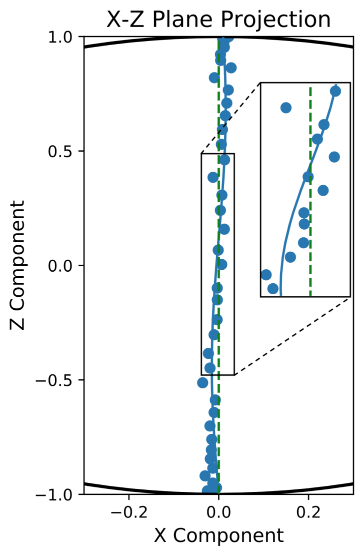
In the preceding subsection, we showed how an augmented gate set can be beneficial. However, the compiler now has more than the minimum set of pulses to work with to realize a quantum gate. In order to decide which pulses to use when, we need a deeper understanding and characterization of the errors incurred by $R_x(\theta)$ gates for arbitrary θ. We can use this system characterization to inform the compiler about the best pulse substitution strategy.

We performed pulse simulations and real experiments to gain insight into the errors. Our simulations were done using Qiskit's OpenPulse simulator. We enhanced the simulator to find the Hamiltonian terms for IBM's Almaden system, through a reverse-engineering process and fitting the results to the device-reported pulse library.

Taking Almaden's pre-calibrated direct X pulse (DRAG pulse), we scaled the area-under-curve down by a factor of $\frac{0}{40}, \frac{1}{40}, \frac{2}{40}, ..., 1$. To first order, these should perform $R_x(\theta)$ for $\theta = 0^\circ, 4.5^\circ, 9^\circ, ..., 180^\circ$. For each angle, we performed three simulations and three experiments to measure the X, Y, and Z components of the final quantum state, which



(a) Sweeping 41 angles from $\theta = 0°$ in green to $\theta = 180°$ in orange.



(b) XZ trajectory slightly deviates from $X = 0$.

Figure 6.6: Simulated results for Direct $R_x(\theta)$. The inset magnifies the X component.

allows us to plot on the Bloch sphere.

Figure 6.6 depicts the results of simulation. Plotting only the X-Z plane, we see that deviations from the Prime Meridian are quite small, but do have a sinusoidal pattern (at exactly 0°, 90°, and 180°, there is no dephasing). These simulation results are in agreement with an independent simulation from [230].

The experimental results are presented in Figure 6.7. We note two deviations from simulation: (1) the X components are still sinusoidal but now translated to the right and (2) the magnitude of the X-component deviations are larger. However, we can treat these characterization results with an empirical attitude—now that we know the dephasing at each θ value point, we can perform an $R_x(\theta)$ gate by applying a scaled-down X pulse, and then correcting the phase error in accordance with the data in Figure 6.7.

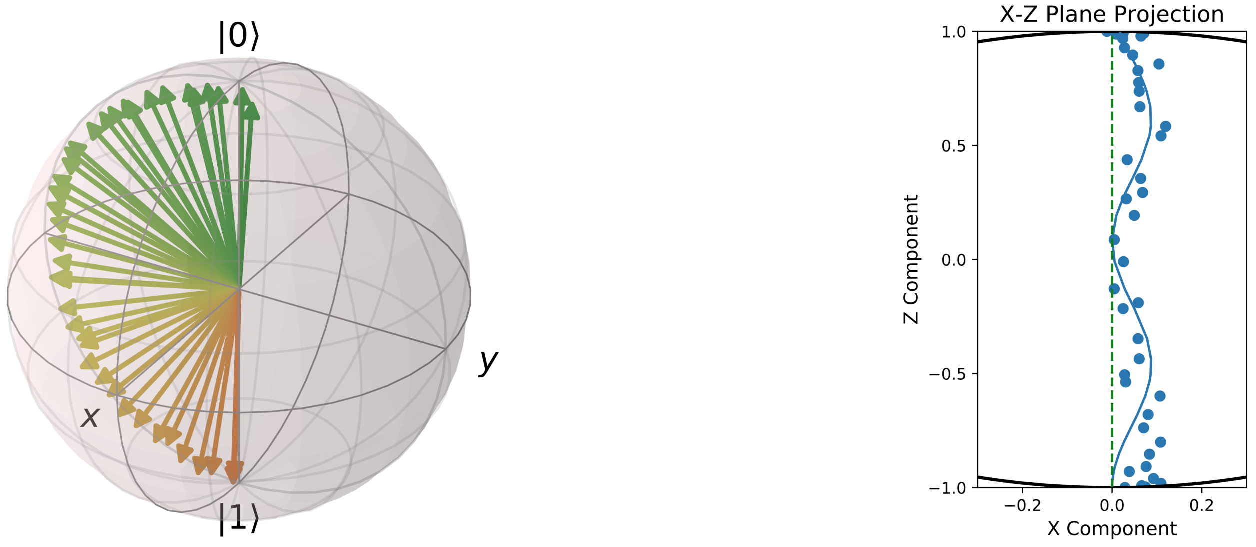


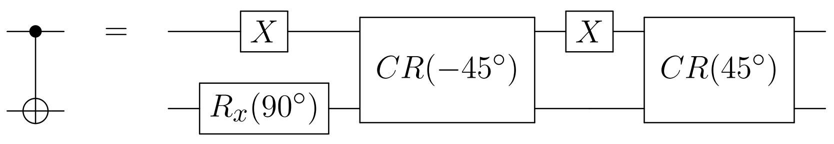
Figure 6.7: Experimental results for Direct $R_x(\theta)$ on IBM's Almaden system, based on $3 \times 41 \times 1000 = 123$k shots. This empirical characterization of dephasing from the Meridian can be used to make the gate better at each θ.

6.5 Optimization 2: Cross-Gate Pulse Cancellation

The gist of this optimization is that standard basis gates are not atomic[3], despite conveying this perception. By augmenting basis gates with the true atomic primitives, new gate cancellation opportunities emerge that lead to 24% speedups for common operations.

6.5.1 Theory

Generally, two-qubit basis gates are not atomic. For example, in Qiskit, the CNOT basis gate is implemented at the pulse level as a combination of single qubit gates, plus invocations of the hardware primitive Cross-Resonance pulse:



3. We use atomic in the common-usage sense of something that cannot be decomposed into something else more fundamental. This should not be confused with technical meanings of atomicity in computing.

Notice in particular, that even the invocation of the hardware primitive Cross-Resonance pulse is not a clean atomic unit, but is decomposed into two pulses separated by an X gate. This "echoed" Cross-Resonance pulse design is necessary to perform a $CR(90°)$ gate (which is the generator of CNOT) with high fidelity [256].

This analysis reveals there are opportunities for gate cancellation on either side of the CNOT[4]. In fact, such sequences are common. To enable these cancellations, we augment the basis gate set with the hardware primitive $CR(\pm 45°)$ basis gates, which are free from pre-calibrated CNOTs. We replace the assembly instruction for CNOT into this decomposition and invoke Qiskit's optimizer to perform gate cancellations.

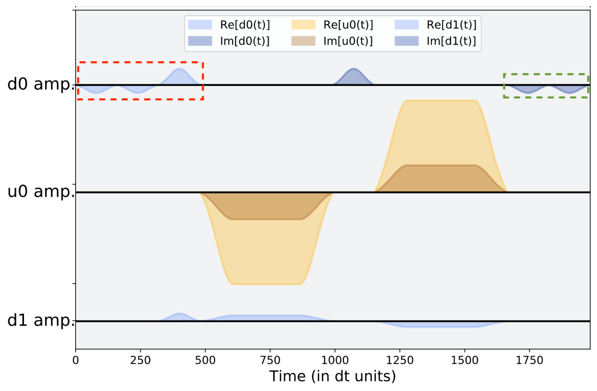
6.5.2 Application

To demonstrate our technique, we benchmarked using a common operation: the open-Controlled-NOT. The open-CNOT has the "opposite" behavior as a CNOT: it flips the target if the control is $|0\rangle$ and does nothing if the control is $|1\rangle$. Its implementation via the CNOT basis gate is simple: first an X on the control, then a standard CNOT, and then another X to restore the control.

However, by decomposing the CNOT into our augmented basis gates, the first X on the control cancels with the "internal" X in the decomposition of CNOT. Figure 6.8 depicts the pulse schedules for the open CNOT under standard compilation (top) and via our compilation (bottom). Notice that two X's in the red box cancel out, leading to a 24% reduction in runtime.

We tested the open-CNOT pulse schedules experimentally. To isolate the effect of cross-gate pulse cancellation, we performed the direct X gate from the previous section in both variants. The resulting data indicates a modest increase in success probability from 87.1(9)%

4. The circuit decomposition clearly depicts gate cancellation opportunities on the left side of the CNOT with the X and $R_x(90°)$ gates; alternatively, the top X can be shifted rightward by commutation identities to create cancellation opportunities on the right side



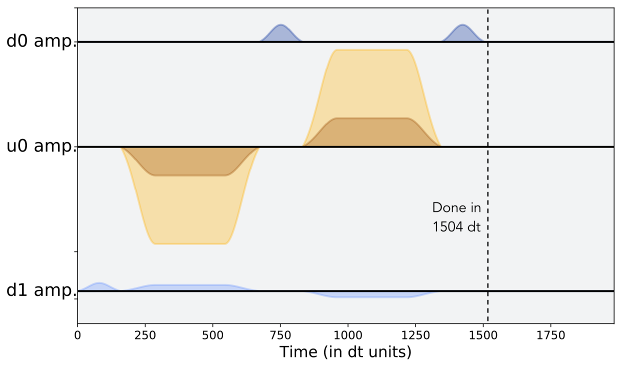


Figure 6.8: Pulse schedules for the open-CNOT by standard compilation (top) and our optimized compilation (bottom). Our compiler cancels out the X rotation gates in the red box and combines the two $R_x(-90°)$ pulses in the green box into a single $R_x(180°) = X$ pulse. This reduces the total duration by 24% from 1984 dt to 1504 dt.

to 87.3(9)%, measured over 16k shots (hence the Bernoulli standard deviation of 0.09%). We emphasize that the open-CNOT is just one of many typical quantum operations that have R_x rotations next to two-qubit basis gates. Our compiler takes advantage of all such cancellation opportunities, which are otherwise invisible at the granularity of standard, non-atomic basis gates.

6.6 Optimization 3: Two Qubit Optimizations

The gist of this optimization is that standard basis gates lead to inefficient decompositions of important two-qubit operations. Instead, we can use pulse-level hardware primitives as new basis gates that lead to operations with 60% lower error.

6.6.1 Theory

Recall from Table 6.2 that two-qubit operations can be achieved by using a "half" or parametrized basis gate set. For example, data movement (SWAP) is 2x more costly on superconducting qubits with an iSWAP basis gate than on qubits with a $\sqrt{\text{iSWAP}}$ basis gate.

Here, we study basis gate decompositions using the parametrized Cross-Resonance pulse $CR(\theta)$, which is the pulse-level hardware primitive on IBM devices. However, we again emphasize that our compiler techniques immediately generalize to any other basis gate decompositions.

As discussed in Section 6.5, neither $CR(\theta)$ nor even $CR(90°)$ are exposed as standard basis gates. Our compiler first extracts the pulse for the $CR(90°)$ gate from the `cmd_def` pulse schedule for the CNOT basis gate. Then, to implement $CR(\theta)$ for arbitrary θ, we horizontally stretch the $CR(90°)$, guided by knowledge of IBM's specific "active cancellation echo" implementation of the Cross-Resonance pulse [255, 257].

Figure 6.9 shows our experimental results, which closely track with the ideal curve. Given the successful implementation of $CR(\theta)$ at the pulse level, we added it as a new basis gate.

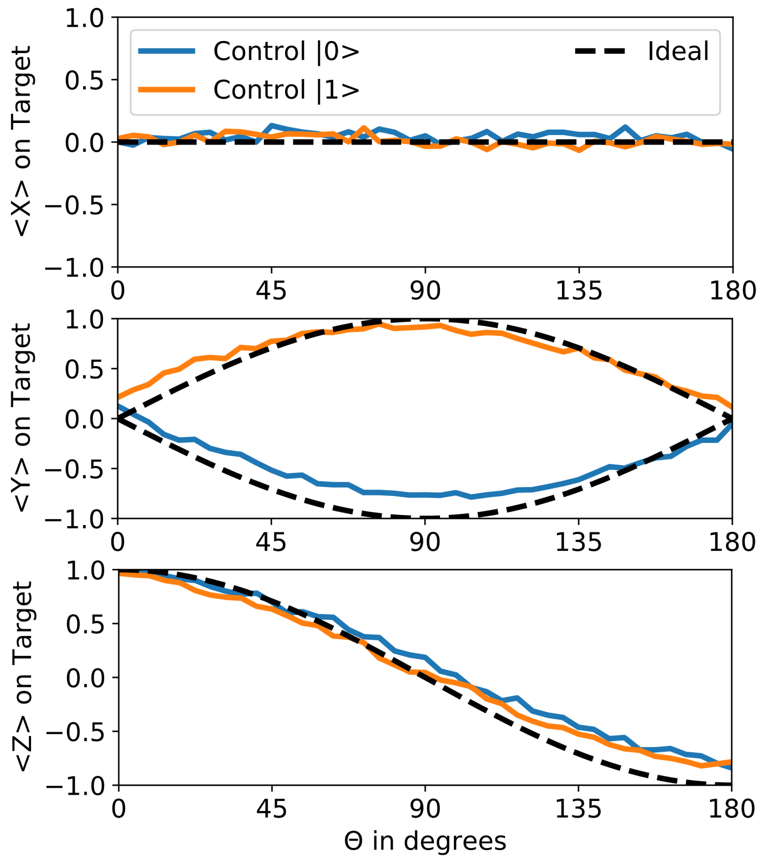


Figure 6.9: Tomography on the target qubit in Cross-Resonance(θ) pulse. Results from both experiment and simulation agree with ideal results. $41 \times 3 \times 2 \times 1000 = 246$k shots.

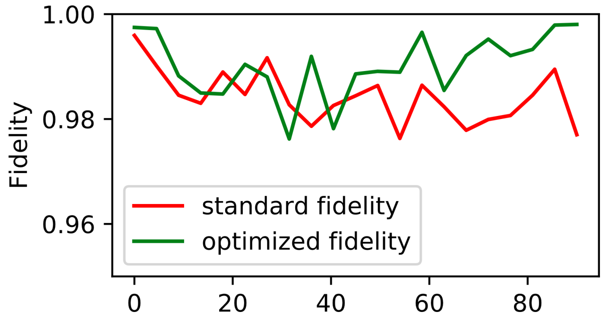
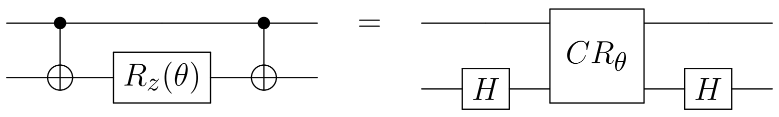


Figure 6.10: Experimental results for state fidelity, measured for the ZZ Interaction by standard compilation vs. our optimized compilation. These results reflect $21 \times 2 \times 2000 = 84$k shots. Standard and optimized have fidelities of 98.4% and 99.0% respectively. Thus, our compiler achieves an average 60% reduction in error for the ZZ Interaction.

6.6.2 Application

As indicated by the last row of Table 6.2, the "ZZ Interaction" two-qubit operation can be implemented using a single CR(θ) gate. By contrast, the "textbook" implementation using standard basis gates requires two CNOTs. The CR(θ) decomposition is depicted below. While this decomposition is fairly simple in hindsight, we discovered it computationally using the optimization procedure mentioned in Section 6.3.2.



To experimentally verify our ZZ Interaction technique, we implemented it using both the standard compiler (i.e. CNOT, $R_z(\theta)$, CNOT) and our optimized compiler (H, CR(θ), H) for θ spanning from 0° to 90° in 4.5° increments. As shown in Figure 6.10, our compiler achieves better results, with a 60% average reduction in error (state infidelity).

As we will see in Section 6.8's Benchmark Results, the ZZ Interaction is the most frequent two-qubit operation in near-term algorithms. Thus, this optimization is the dominant source of improvements in full benchmarks. Before continuing, we re-iterate that our compiler passes (as discussed in Section 6.3.3) automatically identify ZZ Interactions in user-code, even when obfuscated by false data dependencies. Therefore, programmers may continue to write code using “textbook” CNOT decompositions and do not need to reason about device physics.

6.7 Optimization 4: Qudit Operations

The gist of this optimization is that access to quantum hardware at the pulse level enables us to control energy states outside the qubit subspace. In particular, we can instead control our information carriers as d-level *qudits*. We experimentally demonstrate this idea, by cycling a base-3 counter using a single *qutrit*, a task that would be impossible with a single qubit. The counter achieves high fidelity, suggesting practical near-term applications.

6.7.1 Theory

Many quantum systems used to realize a qubit have other energy levels present, which can be used to construct quantum gates [258, 259, 72] or, as we demonstrate in this section, to realize d-level *qudits*. Substantial prior work observed an “information” compression advantage from using 3-level qutrits or higher level qudits [28], which has been further applied to specific algorithms such as Grover search [41, 260, 43, 44] and Shor factoring [45]. More recent work [17] has even demonstrated exponential gains from using qutrits to implement common operations like the Generalized Toffoli.

However, across nearly all quantum hardware and associated software, standard basis gates are only written to address the qubit subspace of hardware. This is the case in part because the local oscillator described in Section 6.3.1 is set to oscillate at the energy gap between the $|0\rangle$ and $|1\rangle$ energy states, f_{01}. Since higher level states are separated by different energy gaps, under normal operation, gates can only address this qubit subspace.

However, we can circumvent this limitation by carefully designing our pulse schedule. For example, suppose we want to address the $|1\rangle$ to $|2\rangle$ transition subspace, whose energy gap we denote as $f_{12} = f_{01} + \alpha$. Per Equation 6.1, applying a $d_j(t) = e^{-i\alpha t}$ pulse yields a total output of $e^{if_{12}t}$. Thus, by designing a frequency-shifting pulse schedule, we can change the effective frequency of the local oscillator and target subspaces beyond the $|0\rangle$ to $|1\rangle$ regime.

6.7.2 Application

By transitioning to these higher energy levels one at a time we can realize a base-d "counter". Not only is this a good benchmark for qudit control, it has potential application in both the near-term era of quantum computing and beyond. In the near-term, parity checks are commonplace [261] (though most parity checks are for even/odd) and a counter (modulo d) serves this exact purpose. Qutrit measurement also enables error mitigation by detecting accidental leakages outside the qubit subspace [262]. Beyond the near-term era, function evaluation oracles are ubiquitous and can be sped up via a counter. For example, recent work demonstrated that just a single qudit, acting as parity check, can implement an oracle-based quantum algorithm [263].

Here we demonstrate the ability to implement a counter via microwave control of a superconducting qubit, using two transitions previously inaccessible by standard basis gates. Specifically, we target the f_{12} and $f_{02}/2$ transitions (bottom right panel of Figure 6.11) which act on the $|1\rangle$ to $|2\rangle$ subspace and the $|0\rangle$ to $|2\rangle$ subspace respectively. The required drive strength and duration for these different transitions are dictated by the inherent coupling between each of the levels of interest, which is determined by the physics of the device. In the case of the two photon $f_{02}/2$ transition, the coupling between the $|0\rangle$ and $|2\rangle$ states is suppressed and thereby requires larger drive powers than those needed for an X gate between $|0\rangle \rightarrow |1\rangle$ transitions, with single photon powers around $p_{\text{one}} \approx 0.109$a.u. and two photon powers of $p_{\text{two}} \approx 0.44$a.u., each 35ns in duration. The f_{12} frequency can be measured either by applying an X gate on the $|0\rangle \rightarrow |1\rangle$ transition and subsequently performing qubit spectroscopy, or by driving a two photon $f_{02}/2$ transition and using the prior knowledge of f_{01} to determine f_{12}. Once the transition frequencies are identified, we calibrate the proper

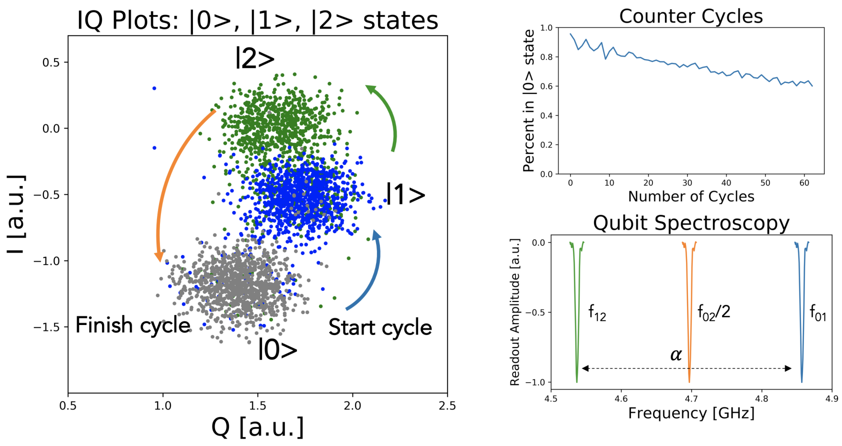


Figure 6.11: (Left Panel) IQ Plot of readout resonator for different quantum states, and the specific cycle we follow. (Top Right) Percentage of shots found in the ground state as a function of the number of cycles. (Bottom Right) Different transition frequencies for the first three energy levels of a superconducting qubit with $2\pi f_{01} \sim 5GHz$ and α ~-300MHz and a two photon transition. These results span 150k experimental shots on IBM Almaden.

amplitude and duration of the pulses to fully switch the qubit to the desired final state.

To gauge the fidelity of our counter, we start off by training a linear discriminator to identify the qutrit state upon readout. In the case of this work, we train a sklearn [264] Linear Discriminant Analysis classifier with the calibrated qutrit $|0\rangle, |1\rangle, |2\rangle$ states and corresponding resonator IQ values (left panel of Figure 6.11). Once these calibrations are made, we measure the percentage of shots that have the qutrit in the $|0\rangle$ state at the end of the cycle. Due to imperfections in microwave control, our results deviate from the ideal of 1.0 as the number of cycles increase, making this an ideal testbed for further research such as improved microwave control [265, 266] and optimal readout parameters [267]. Nonetheless, the results indicate remarkably high fidelity—we can drive 60 cycles or 180 hops, before "dropout" exceeds 40%. This appears promising for counting or parity check applications.

6.8 Results and Discussion

6.8.1 Benchmarks

We applied our compiler towards full quantum algorithms. Before proceeding, we note two thematic differences between our treatment of experimental benchmarks and that done in recent architectural work.

First, we focus exclusively on near-term algorithms. Some recent work [268, 212, 269, 213, 6] demonstrated impressive compiler optimizations for algorithms like Bernstein-Vazirani [270], Hidden-Shift [271], Adders, and Quantum Fourier Transform [272]. However, we emphasize that these algorithms are not representative of near-term algorithms, which are generally based on a *Hamiltonian simulation* kernel that quantum computers can naturally compute efficiently. Hamiltonian simulation, and thus near-term algorithms broadly, are dominated by the ZZ Interaction optimized in Section 6.6. We specifically evaluated three types of near-term algorithms: (1) Variational Quantum Eigensolver (VQE) [84], which addresses minimum-eigenvalue problems such as molecular ground state estimation; (2) Quantum Approximate Optimization Algorithm (QAOA) [85], which approximates solutions to NP-Hard combinatorial optimization problems; and (3) Hamiltonian Dynamics, which models molecular dynamics and was recently adapted for near-term applications [273, 274].

Second, we use Hellinger error/distance (or its complement, Hellinger fidelity) as our top-level metric. Intuitively, Hellinger error captures the distance between two probability distributions: two identical distributions have the minimum distance of 0 and two completely antipodal distributions have the maximum distance of 1. Often, it is appealing to use Probability of Success (i.e. of finding the MAXCUT) as the top-level metric for algorithms like QAOA-MAXCUT [275, 276]. However, QAOA is not intended to find the MAXCUT with 100% successful probability (otherwise it would solve NP-hard problems in polynomial time), so a QAOA experiment with 100% "Probability of Success" would actually reflect

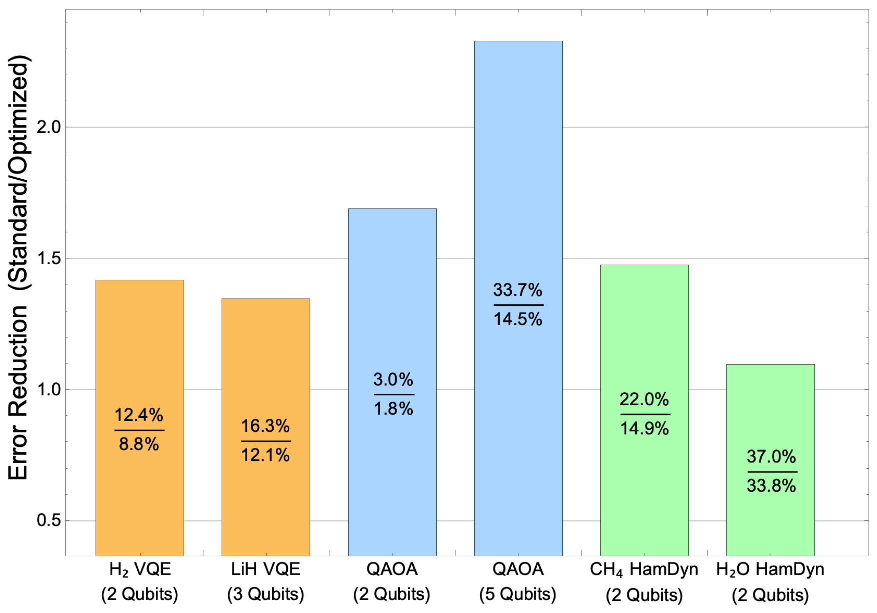


Figure 6.12: Reduction in error (Hellinger distance) for benchmarks, due to our optimizations. These results reflect $6 \times 2 \times 8000 = 96$k shots on IBM Almaden.

high error. Instead, QAOA is intended to compute a *distribution* of measurement outcomes, within which bitstrings with large cuts will have boosted probabilities. This motivates our use of Hellinger error, and we urge subsequent experimental work to also evaluate near-term algorithms on the basis of probability-distribution distances.

6.8.2 Results

Figure 6.12 shows the reduction in error due to our optimizations. The H_2 and LiH VQE benchmarks replicate recent experimental work, [277] and [151] respectively. Both experiments are based on the Unitary Coupled Cluster ansatz [114]. The QAOA benchmarks compute MAXCUT on an N-qubit line graph. The Hamiltonian dynamics simulation benchmarks both simulate 6 Trotter steps. The methane and water Hamiltonians were generated with OpenFermion [143], taking advantage of orbital reductions to reduce the problems to two qubits.

For all six benchmarks, our optimized programs run with much lower error (Hellinger distance/infidelity) between the actual and target outcome distributions. The average error reduction factor is 1.55x and the largest benchmark, 5 Qubit QAOA, has a 2.32x reduction in error from 33.7% to 14.5%. The majority of our error reduction stems from our optimization of the ZZ Interaction by augmenting the basis gates with direct access to the Cross-Resonance pulse. We focus on Hellinger error because it is accepted in the quantum community as an (in)fidelity metric and has a "linear" interpretation. The average 1.55x error reduction factor is comparable to a year worth of hardware progress; of course, our method is achievable now and is performed in software. Similar work for QAOA was also recently demonstrated on Rigetti's hardware, using a parametrized ZX interaction [278].

In addition to the six qubit benchmarks, we also ran the qutrit incrementer in Section 6.7 and demonstrated 60 cycles, i.e. 180 increment operations, before "dropout" exceeds 40%. This benchmark is unique, because it has no standard qubit comparison—a single qubit cannot model a base-3 counter. This high-fidelity qutrit control confirms that pulse-backed basis gates offer a promising path towards qudit-based optimizations.

6.8.3 Source of Fidelity Improvements

The fidelity improvements presented here have three sources:

1. **Shorter pulses**. Our compiler's optimized pulses are shorter: 2x shorter for the single qubit rotations in Optimization 1, 24% shorter for open-CNOTs due to Optimization 2, and ~2x shorter for ZZ interactions due to Optimization 3. These lower operation latencies are advantageous because qubits have less time to decohere.

2. **Less calibration error susceptibility**. `DirectRx(`θ`)` only applies one pre-calibrated (and then amplitude-downscaled) pulse. By contrast, the standard decomposition applies *two* pre-calibrated pulses, squaring the impact of calibration imperfections.

3. **Smaller pulse amplitudes**. Our pulse shaping techniques either vertically downscale

amplitudes (Optimization 1) or horizontally stretch pulses (Optimization 3). As such, our pulse amplitudes are smaller than or equal to those generated by standard compilation. This is beneficial because smaller pulse amplitudes have smaller spectral components, reducing leakage to undesired frequency sidebands—see Figure 14 in [230] for details.

Our experience indicates that all three of these sources have meaningful contribution to the fidelity improvements. To further understand our fidelity improvements and reduce the impact of State Preparation and Measurement errors, we performed a Randomized Benchmarking [279] style experiment. In the experiment, we select $K-1$ random single-qubit unitary operations. We execute these $K-1$ operations, terminated with 1 final single-qubit operation that inverts all of the preceding operations. Therefore, under noise-free execution, the qubit returns to the initial state of $|0\rangle$ with 100% probability. However, due to noise, error accumulates as we increase K from 2 to 25.

Figure 6.13 presents our results, which ran over several hours on IBM's Armonk device. The *optimized* plot results from compiling with Optimization 1: Direct Rotations. However, to isolate the effect of shorter pulses, we also compiled *optimized-slow*, which inserts NO-OP idling into the optimized pulse schedules, to match the duration of the standard pulse schedules.

Each trajectory was fit to the exponential decay, $f^K - b$, where b is a y-intercept term that represents SPAM errors independent of K, and f is interpreted as gate fidelity. The resulting gate fidelities for optimized, optimized-slow, and standard are $f = 99.87\%$, 99.83%, and 99.82%. This implies that shorter pulses (#1) account for 70% of the fidelity improvement, while less susceptibility to calibration imperfection (#2) and smaller pulse amplitudes (#3) account for the remaining 30% improvement. The improvement due to shorter pulses matches theoretical predictions: according to the *gate error in coherence limit* calculation [76, Eq. 24], the 2x pulse speedup yields a minimum 0.01% fidelity improvement.

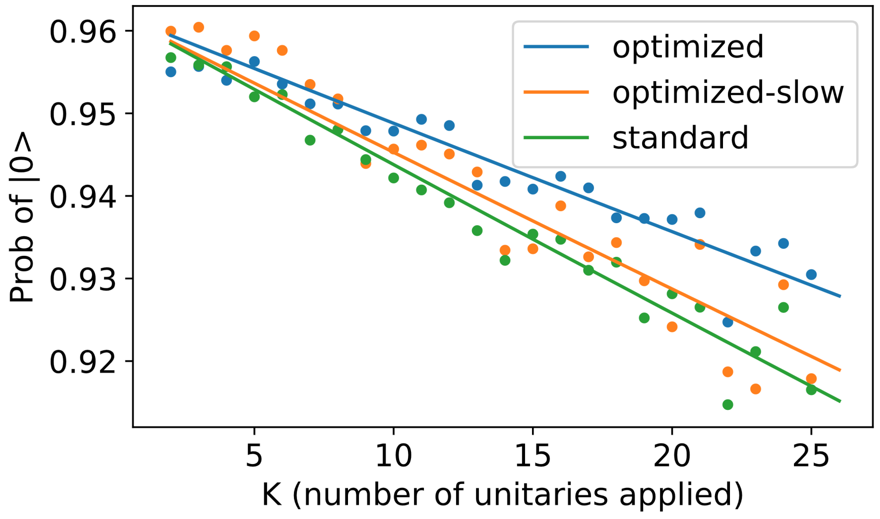


Figure 6.13: Randomized Benchmarking style experiment, fit to exponential decay. For each K, we randomized 5 sequences of unitary operations. $5 \times 24 \times 3 \times 8\text{k} = 2.88\text{M}$ total shots.

6.9 Conclusion

Our results demonstrate that augmenting basis gates with pulse backed hardware primitives, bootstrapped from existing calibrations, leads to 1.6x error reductions and 2x speedups for near-term algorithms. Critically, our technique does not rely on knowledge of the system Hamiltonian, thus bypassing the experimental barriers to quantum optimal control. The measured fidelity improvements are arguably equivalent to a year's worth of hardware progress, but our techniques are available immediately, through software. We hope that our experiences with OpenPulse will encourage more quantum vendors to expose their hardware to pulse-level control. To this end, all of our code and notebooks are available on Github [210].

CHAPTER 7

OPTIMIZATIONS WITH FAN-OUT IN NEAR-TERM QUANTUM COMPUTERS

7.1 Introduction

Instruction scheduling is a powerful compiler technique in both classical and quantum computing. In the classical realm, scheduling techniques such as pipelining, Single Instruction Multiple Data (SIMD), and Out-of-order execution have led to continued gains in processing power. These scheduling techniques are designed to preserve a program's logical correctness by respecting constraints known as *hazards*.

Just as in the classical setting, quantum computing is also amenable to instruction scheduling. In fact, due to the short lifetimes of qubits in the NISQ (Noisy Intermediate-Scale Quantum) era [16], scheduling to reduce latency is critical for successful execution [280, §II. E.]. The potential of quantum instruction scheduling was recently exemplified by Google's Quantum Supremacy result [14], which experimentally demonstrated a task soluble in seconds on a 53 qubit computer that is argued to likely require days [15] on a supercomputer. At the core of the Supremacy result is a *coupler activation* [14] schedule that maximizes simultaneous resource utilization.

A number of papers [27, 281, 282, 283] in the architecture community have studied quantum scheduling, inspired by techniques from the classical setting. One principle underlying these papers is **exclusive activation**: a qubit can be involved in at most one operation per timestep [27]. In architectural terms, this is a *structural hazard* [284]. Under exclusive activation, schedulers optimize for data parallelism by simultaneously executing instructions on disjoint qubits. However, there are natural limits to such schedulers, since instructions on overlapping qubits must be serialized.

Our work begins with a simple but consequential observation: the structural hazard of exclusive activation is not actually enforced by most quantum hardware. In fact, it is often *more* natural for a quantum processor to simultaneously execute multiple operations on shared qubits through *global interactions*. The building block of our work is the fan-out operation depicted in Figure 7.1. This operation can be understood purely classically. The four CNOT (Controlled-NOT) gates at the left each comprise a control (•) and a target (⊕), and the target is flipped iff the control qubit is 1. This operation performs fan-out for classical input states: when the targets are initialized to 0, the control bit gets copied to the targets.

While exclusive activation would serialize the four CNOT instructions as depicted on the left, underlying quantum hardware can naturally perform these interactions simultaneously, as depicted on the right. This form of Single Instruction Multiple Data (SIMD) parallelism arises only after discarding structural hazards that don't manifest in hardware. As we demonstrate later, the fan-out building block generalizes to efficiently-scheduled circuit synthesis for the ubiquitous Controlled-U operation. Henceforth in this chapter, fan-out will refer to simultaneous operation on the right of Figure 7.1.

We begin in Section 7.2 by surveying relevant prior work. The three subsequent sections capture our core contributions:

- Section 7.3: We generalize the simultaneous fan-out primitive into a **circuit synthesis procedure to schedule Controlled-U** operations with an asymptotic depth advantage.

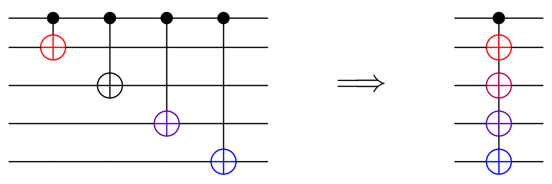


Figure 7.1: Device level fan-out allows a NOT to the bottom four targets iff the top control is on. While exclusive activation induces serialization (left), quantum hardware can implement fan-out simultaneously (right) in a single step.

- Section 7.4: We leverage this circuit synthesis procedure to **optimize NISQ circuits** (which rely on Controlled-U). We also introduce **novel quantum memory architectures**.
- Section 7.5: We perform **technology modeling** of simultaneous fan-out on trapped ion qubits.

Section 7.6 presents results for several benchmarks. Section 7.7 proposes an implementation of fan-out on superconducting qubits and demonstrates experimental proof-of-concept. Sections 7.8 concludes. To maintain a focus on architectural aspects, we omit details about benchmarks and hardware physics. For readers without prior quantum computing experience, we also provide sufficient background in Section 7.9 so that this chapter is self-contained. Finally, Section 7.10 has further discussions regarding the scalability of the superconducting fan-out procedure in Section 7.7.

7.2 Prior Work

Our work builds on top of prior work from the (1) computer architecture, (2) computer science theory, and (3) physics communities. At a high level, the priorities of the work in each community can be characterized as follows:

1. architects have devised intelligent schedulers/circuit synthesis tools, but they assume a false structural hazard by overlooking global interactions
2. theorists have devised intelligent circuit constructions assuming global gates, but they don't consider NISQ workloads or device-level operation
3. physicists have studied global interactions, but usually in an ad hoc fashion separated from computation and NISQ workloads

Our work unites insights from all three disciplines to devise a circuit synthesis tool that leverages global interactions to accelerate NISQ workloads.

7.2.1 Computer Architecture

Amongst architects, a number of papers [27, 281, 282, 283, 60, 285] have studied instruction scheduling in quantum computers. These papers all assume some structural hazard against simultaneous execution of overlapping qubits. [27, 281] provides the most formal description of this hazard, terming it as the principle of exclusive activation which forbids a qubit from being involved in more than one operation per timestep. Moreover, hardware-dependent considerations such as crosstalk [286, 287] further narrow the scope of when operations can be parallelized. For example, on superconducting hardware, `CNOT(a,b); CNOT(c,d);` may be forbidden simultaneously, even though the `CNOT` gates are disjoint.

In other architectural work such as [282] and [285], the authors provide examples for obtaining data parallelism on disjoint instructions. However, in both papers, the examples ultimately incur serialization upon encountering gates on overlapping qubits. As we will demonstrate in Section 7.3, this serialization is unnecessary.

Finally, [283] describes exclusive activation as a data dependency, since the no-cloning theorem [288] prevents copying a qubit to participate in multiple instructions simultaneously. This is indeed a valid perspective. Regardless, we will demonstrate that the underlying problem is in fact addressable with the fan-out primitive.

7.2.2 Computer Science Theory

Quantum fan-out has also been studied from a complexity theory lens. [289] proved that the $\mathrm{QNC^0}_\mathrm{f}$ circuit class with unbounded fan-out is powerful for fault-tolerant applications such as Shor's factoring algorithm [10]. Other applications of fan-out to arithmetic operations such as addition, OR, and modulus are considered in [290, 291, 292]. Finally, [293] shows that under widely-held complexity theory assumptions, fan-out in quantum circuits can increase the hardness of classical simulability. Our work revisits these theory results with NISQ workloads and underlying device physics in mind.

7.2.3 Physics

The engineering of global interactions on N qubits has been well studied in device physics communities. A common benchmark for global interactions is the preparation of the *GHZ state* [294], a task which is essentially equivalent to fan-out. Experimentally, global interactions have been used to prepare the GHZ state on a variety of leading qubit technologies including Trapped Ion [295], Neutral Atom [206], and NMR [296]. Implementation on NV center qubits has been proposed as well [297]. Notably, superconducting qubits, which are the current leader in hardware scale, were not previously known to support simultaneous overlapping interactions. However, in Section 7.7, we experimentally demonstrate simultaneous fan-out on superconducting qubits.

Global interactions have already been noted by physicists for their application to Hamiltonian simulation, an important quantum algorithm. As early as 2005, [298] noted that global interactions enable constant depth parity measurement, an important building block for Hamiltonian simulation. Later work [299, 227] further optimized the procedure. Recently this year, three papers [300, 301, 302] have applied global interactions to building blocks of longer-term fault tolerant quantum computers. These papers demonstrate that the Generalized Toffoli operation can be performed in constant time with global interactions, whereas otherwise linear or log depth is required [17].

Very recent papers have adopted an interdisciplinary approach, combining insights from physics and architecture. For example, [227]—which inspired our work—describes fan-out as SIMD parallelism. Also, a recent trapped ion hardware paper [303] describes global interactions as a form of Multiple Instruction Multiple Data (MIMD) parallelism. Our work continues this architectural perspective, while also focusing on NISQ circuit optimizations and further refining the underlying technology models based on recent experimental developments.

7.3 Controlled-U Synthesis

The basic building block of our work is the simultaneous fan-out operation depicted on the right side of Figure 7.1. Two important considerations arise in evaluating the applicability of fan-out. The first is whether the simultaneous implementation via global interactions truly achieves a linear speedup over serialized CNOTs. As described in Sections 7.5 and 7.7, experimental results from hardware assert this is indeed the case. The second consideration is how fidelity is affected by simultaneous fan-out versus serialized CNOTs. Our results in Sections 7.5 and 7.7 indicate a modest improvement in fidelity from simultaneous fan-out.

We focus on a circuit synthesis procedure that uses fan-out to optimize the Controlled-U operation, described below. This operation is ubiquitous in NISQ algorithms, and each application in Section 7.4 is an instance of Controlled-U. As we will describe in this section, our circuit synthesis procedure yields a Controlled-U implementation that is scheduled to align CNOT gates into a single fan-out step. This yields asymptotic improvements in circuit depth.

The controlled-U operation is depicted at the left of Figure 7.2. As in other controlled operations like CNOT, the U operation should be applied if and only if the top control qubit is $|1\rangle$. However, unlike the single-qubit U in Figure 7.14e, here we consider the case when U is an operation on multiple qubits. Therefore, U itself has a decomposition into gates, shown under the blue overlay. Our results are applicable for any decomposition basis, but we focus on the decomposition into the universal set of single-qubit + CNOT gates, since quantum algorithms are typically expressed in this form. In the example, U has a width of four qubits and a depth of two layers. The first layer contains four disjoint single qubit gates, and the second layer contains two disjoint CNOTs.

Under exclusive activation, implementation of Controlled-U is bottlenecked by the dependence of each controlled gate on the single control qubit. Thus, the parallel two-layer implementation of U collapses into a serial implementation of Controlled-U as depicted at the right of Figure 7.2. The amount of serialization is proportional to the width of U, so that the effective depth of a Controlled-U operation under serialization is $O(\text{Depth} \times \text{Width})$. In many workloads, the width greatly exceeds depth, so this serialization is very harmful.

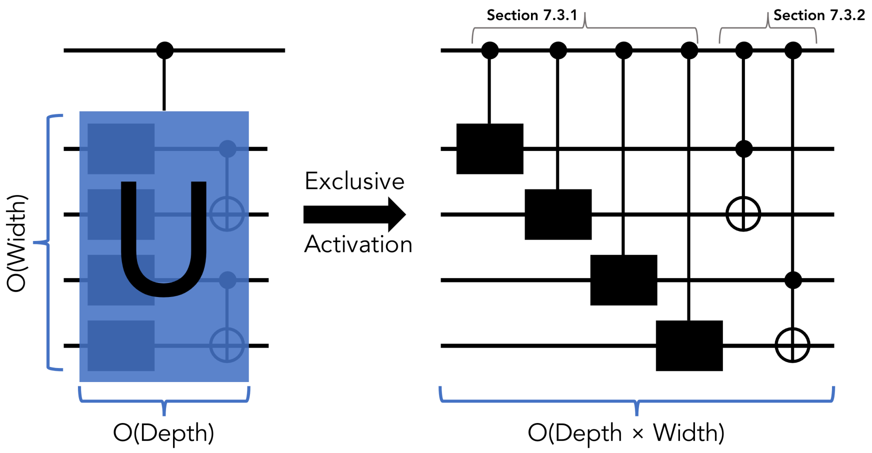


Figure 7.2: Left: general form of controlled-U. Right: under exclusive activation, adding the control induces serialization and multiplies the effective Depth by the Width.

It is not immediately obvious how fan-out can help speed up Controlled-U. Whereas fan-out is a SIMD operation, Controlled-U is a MIMD operation, since the gates in U are arbitrary. However, we can resolve this difficulty be decomposing gates into a form amenable to 'alignment' of CNOTs into a single fan-out step. This circuit synthesis procedure has two underlying cases. The first, Shared-Control Single Qubit Gates, supports the simultaneous execution of multiple Controlled-U_i gates with a shared control qubit. This procedure applies to the first layer of U in Figure 7.2. The second, Shared-Control Toffoli's, supports the simultaneous execution of multiple Controlled-CNOTs with a shared control qubit. These double-controlled NOTs are referred to as Toffoli's. The Shared-Control Toffoli's case applies to the second layer of U in Figure 7.2.

In practice, arbitrary U's will also contain mixed layers that contain both single-qubit gates and CNOTs. This general case can be handled by unifying the synthesis procedures for Shared-Control Single Qubit Gates and Shared-Control Toffoli's.

Table 7.1 compares the time (depth) and space (ancilla qubits) costs of implementing

Controlled-U. Our work, which uses fan-out, is optimal with $O(D)$ depth (and very small constants) and 0 ancilla qubits. The status quo approach of serialization incurs $O(ND)$ depth which is harmful because $N >> D$ in many applications. Past work in [289] and [304] has proposed alternative approaches for parallelizing circuits using global interactions. In the best case, where a "basis-change" is cheap and efficiently computable, [289] matches our $O(D)$ depth. However, it is extremely expensive in space, requiring $O(N^2)$ ancilla qubits. Finally, [304] provides a numerical optimization technique for compiling Controlled-U down to the minimal possible depth. In this sense, it could achieve the $O(D)$ lower bound. However, the numerical optimization for compilation has exponential cost—simply defining the optimization problem involves specifying a $2^N \times 2^N$ sized matrix. Moreover, the optimization itself is expensive, and convergence to $O(D)$ depth is not guaranteed.

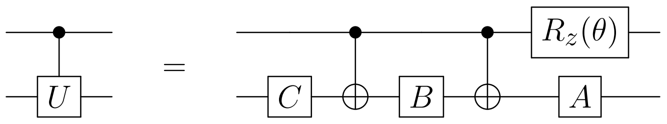
	Depth	Ancilla Qubits
Our Work (with fan-out)	$\mathbf{O(D)}$	**0**
Serialization	$O(ND)$	0
[289] (if cheap basis-change)	$O(D)$	$O(N^2)$
[304] ($\Omega(2^N)$ compile time)	$O(D)$?	0

Table 7.1: Cost of implementing a controlled-U operation in time (depth) and space (ancilla qubits). U has a depth of D and width of N qubits.

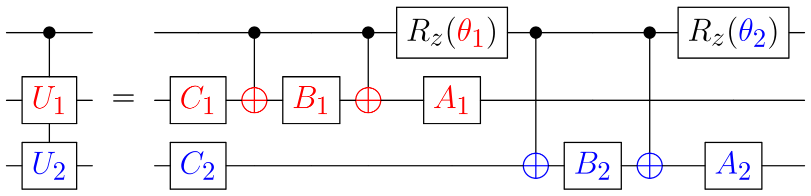
While our procedure achieves the best possible asymptotic space and time costs, it is not as general as [289] and [304]. Our procedure only addresses the special case of Controlled-U parallelization, whereas [289] and [304] apply to the parallelization of any commuting gates or the depth reduction of any unitary, respectively. Nonetheless, our specialization is justified because the Controlled-U template is ubiquitous in NISQ workloads.

7.3.1 Shared-Control Single Qubit Gates

Here, we consider how to simultaneously execute controlled single-qubit gates with a shared control, as in the first layer of U in Figure 7.2. This is a form of MIMD parallelism with overlapping data, but we only have access to the fan-out SIMD primitive. However, we can make progress by invoking the following well-known identity [211] for decomposing controlled single-qubit gates. It shows that for any single-qubit gate U, the Controlled-U operation can be implemented by using CNOT as the only two-qubit gate. Specifically there exist (trivially computable) single-qubit gates A, B, C, and an angle θ, such that



Let us consider applying this identity to a small example: attempting to parallelize Controlled-U_1 and Controlled-U_2 targeting two different qubits. The result is shown below, with colors used for disambiguation.



It appears that applying the circuit identity led to minimal improvements—only C_2 can slide left to execute simultaneously with controlled-U_1. The rest of the blue gates are unable to parallelize, because they are blocked by an apparent dependency on the $R_z(\theta_1)$ gate. However, recalling the commutativity rule in Figure 7.16b, we see that the apparent dependence of the blue CNOTs on the $R_z(\theta_1)$ is actually a false dependence. By commuting the $R_z(\theta_1)$ gate to the end of the circuit, we attain the final result in Figure 7.3.

We have now demonstrated simultaneous execution of shared-control U_1 and U_2 on overlapping data (top+middle and top+bottom qubits respectively), using the fan-out primitive. This pattern extends *ad infinitum* to more qubits—the total depth will always consist of five layers: two fan-out layers and three single-qubit gate layers. For certain gates, the cost could be reduced even further. For instance, for $U = Z$, it is known that the

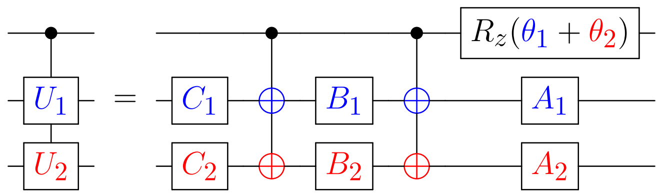
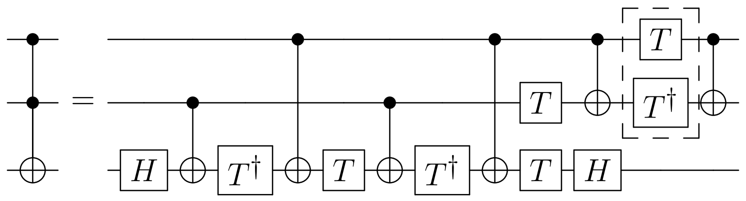


Figure 7.3: Simultaneous execution of shared-control single qubit gates, using the fan-out primitive. This decomposition has constant (5 layer) depth, independent of width.

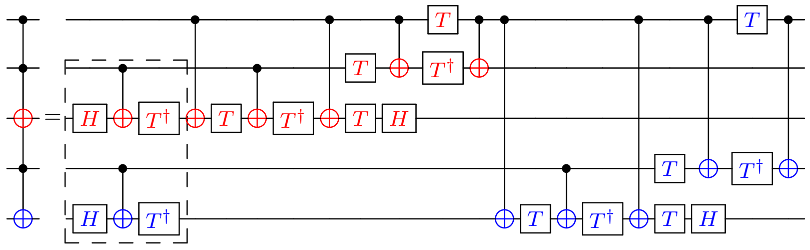
Controlled-Z operation can be implemented with just a single CNOT [211].

7.3.2 *Shared-Control Toffoli's*

The second piece needed for optimized Controlled-U synthesis is simultaneous execution of shared-control Toffoli's. Here, we seek to simultaneously execute multiple Toffoli (Controlled-CNOT) gates, where the CNOTs are disjoint but the additional control is shared across the CNOTs, as in the second layer of U in Figure 7.2. Since Toffoli is a three-qubit operation, it must first be decomposed into single-qubit gates and CNOTs. The standard [211] decomposition is shown next. T and $T^\dagger$ are shorthand for $R_z(\frac{\pi}{8})$ and $R_z(\frac{-\pi}{8})$ respectively.



The boxed group with T and $T^\dagger$ is one example of data parallelism. This level of data parallelism is referred to as a coarse-grained schedule in past architectural work [285]. Next, let us consider applying the Toffoli decomposition to a small example: attempting to simultaneously execute two shared-control Toffoli's, where the CNOTs are disjoint. This exact example is also considered in Figure 4 of [285]. The result is shown below, with colors used again for disambiguation.



As indicated by the boxed layers, only three gates from the blue Toffoli were able to parallelize with the execution of the red Toffoli. This level of parallelization, which results in 21 layers of depth, is referred to as fine-grained scheduling in [285]. While it is slightly better than coarse-grained scheduling, it still linearly serializes the depth. However, we can again leverage commutativity relationships to proceed further and exploit our fan-out primitive.

Notice that the dependency between the right-most red CNOT and the subsequent blue CNOT is in fact a false dependency. These two gates commute per the rule in Figure 7.16a since their targets are different. After transposing the two gates, we encounter a T gate that commutes with the control of the blue CNOT, per the rule in Figure 7.16b. Repeating such commutative transpositions, we can push the blue CNOT to the left to align into a single fan-out. The rest of the blue circuit can be handled similarly, resulting in the final form presented in Figure 7.4. Since $T = R_z(\frac{\pi}{8})$, the $T \times T$ gate at the top right is just a single $R_z(\frac{\pi}{4})$ gate.

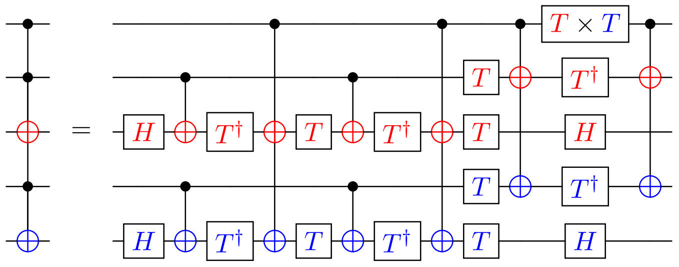


Figure 7.4: Simultaneous execution of shared-control Toffoli's using the fan-out primitive. This decomposition has constant (12 layer) depth, independent of width.

The design in Figure 7.4 extends naturally to more qubits. Regardless of the number of qubits, the depth of the circuit is always 12 layers. Since the depth of a single Toffoli operation is also 12 layers, this means that our shared-control Toffoli's synthesis is optimal. For the circuits we will encounter in the following sections, the number of Toffoli's spans the entire circuit. Therefore the depth cost of the other approaches is $O(N)$, versus our $12 = O(1)$ constant depth.

The combination of simultaneous shared-control single qubit gates and Toffoli's enables a depth-optimized execution schedule for any Controlled-U. Moreover, the multiplicative constants for our circuit synthesis are small. Shared-control single qubit gates incur a depth of just 5 layers, which matches worst case depth. Shared-control Toffoli's incur no depth expansion relative to a single Toffoli and are thus optimal. The resulting Controlled-U circuit synthesis procedure is implemented in `controlled_u.py`. In the following section, we apply the Controlled-U procedure to optimize several NISQ-important quantum circuits, which are all fundamentally Controlled-U operations. While our approach is already asymptotically optimal with low constants, in some cases we can reduce the depth constants even further. This is exemplified by the SWAP Test, which we discuss next.

7.4 Applications

We now examine how Controlled-U circuit synthesis can be leveraged to optimize NISQ circuits. We also apply fan-out to develop novel quantum memory architectures. Table 7.2 summarizes the spacetime advantages of our work (using simultaneous fan-out) for the applications surveyed in this Section.

7.4.1 SWAP Test

One of the most important [308] procedures in quantum computing, especially NISQ machine learning algorithms, is the calculation of inner products between quantum states. This inner product reports the *overlap* or similarity between states. For two qubit registers $|A\rangle$ and $|B\rangle$, this overlap is denoted by $|\langle A|B\rangle|^2$. For equal states $|\langle A|B\rangle|^2 = 1$, and for orthogonal

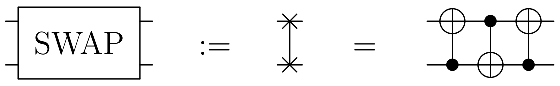
Application	Spacetime costs
SWAP Test between two $k = \frac{N-1}{2}$ qubit registers	(0 ancilla for all)
Our work	$14 = O(1)$ depth
Serialized	$\sim 14k = O(N)$ depth
Coarse-grained sched. [289]	$\sim 12k = O(N)$ depth
Fine-grained sched. [289]	$\sim 9k = O(N)$ depth
Hadamard Test; N-qubit circuit; U has depth D	
Our work	$O(D)$ depth, 0 ancilla
Other approaches (Table 7.1)	$O(ND)$ depth, $O(N^2)$ ancilla, or $\Omega(2^N)$ compile time
Explicit Memory with n index qubits and bitwidth W	
Our work	$O(n)$ depth, 0 ancilla
Bucket-Brigade QRAM [305]	$O(W2^n)$ depth, 0 ancilla
Parallel QRAM [306]	$O(Wn)$ depth, $O(2^n)$ ancilla
Implicit Memory with n index qubits and bitwidth W	($\sim 1 \cdot n$ ancilla for both)
Our work	$O(2^n)$ depth
QROM [307]	$O(W2^n)$ depth

Table 7.2: Summary of space (ancilla qubits) and time (depth) costs for different applications. Our work leverages the ubiquity of simultaneous fan-out to attain asymptotic advantages.

states $|\langle A|B\rangle|^2 = 0$.

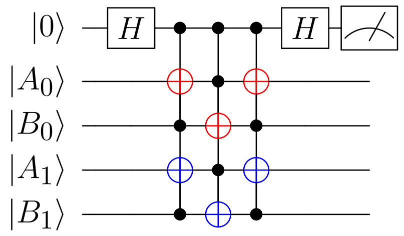
The calculation of this overlap is a procedure known as the SWAP Test. The SWAP Test features heavily in NISQ applications such as quantum kernel classification, which was introduced in [309] and realized experimentally on IBM's quantum hardware in [310]. These quantum kernel methods are noise resilient and amenable to noise mitigation [310]. Further work [311] has introduced kernels that have strong complexity theory foundations for hardness of classical simulability. All of these kernel methods require the evaluation of inner product overlaps. The SWAP Test is also integral to cost function evaluation in NISQ-friendly deep quantum neural networks [312]. In the near-term (and in fact current-term), experimental sequences in quantum sensing [313] are essentially overlap measurements.

The SWAP Test has a very simple form. It is essentially just the case of Controlled-U with $U = \text{SWAP}$. First, we examine the decomposition of a SWAP between two qubits:



This decomposition is equivalent to the triple XOR sequence for in-place SWAPs of classical bits. For a SWAP Test, we need to perform this $U = \text{SWAP}$ sequence not just between two individual qubits, but between two registers of qubits. Moreover, the SWAP is controlled on an ancilla qubit. The SWAP Test also requires a Hadamard-sandwich around the controls, and a measurement of the ancilla. After executing such a circuit, the overlap between the two registers is related by a simple function to the probability of measuring $|0\rangle$ on the ancilla. Repeated executions can therefore estimate the overlap to a desired precision.

Let us concretely consider the example of a SWAP Test on two two-qubit registers, $|A = A_1A_0\rangle$ and $|B = B_1B_0\rangle$. To disambiguate the gates, we have used colors and interleaved the bit ordering of the $|A\rangle$ and $|B\rangle$ registers below:



Under standard serialization of the shared-control gates, the depth is 63 at best from fine-grained scheduling. However, our Controlled-U synthesis procedure, specifically the shared-control Toffoli's decomposition, is directly applicable here. The resulting SWAP Test depth is $3 \times 12 = 36$ (ignoring the two Hadamard gates). Moreover, our procedure always yields a constant depth of 36 layers regardless of the circuit width N, whereas serialized approaches scale as $O(N)$.

While this asymptotic advantage is already appealing, we can attain even further cost reductions to our constants via a circuit identity. It can be shown that the outer two controls on the ancilla qubit can be removed [314, 211]. After this optimization, the final circuit has a depth of just 14 layers, regardless of the size of the SWAP Test.

Interference Circuit

Recent work has explored alternatives to the traditional SWAP Test, with the aim of reducing spacetime costs. The most promising one is the interference circuit [315, 308], which halves the qubit width requirement. Whereas the traditional SWAP Test requires $2k + 1$ qubits to compute the overlap of two k-qubit registers, the interference circuit only requires $k + 1$ qubits. In order to use the interference circuit, we must know the sequences of gates U_A and U_B that can create $|A\rangle$ and $|B\rangle$, respectively. In practice, this is indeed the case for useful applications. The interference circuit has the following simple form shown in Figure 7.5. As in the traditional SWAP Test, the overlap is a simple function of the probability of measuring $|0\rangle$ on the ancilla.

The open-control (open circle) on U_B activates on $|0\rangle$ and can therefore be replaced with an ordinary control surrounded by NOT ($\oplus$) gates. Therefore our Controlled-U is directly applicable to the interference circuit, and it allows overlap calculation with no asymptotic depth overhead relative to U_A and U_B. This is again a linear $O(N)$ speedup via fan-out.

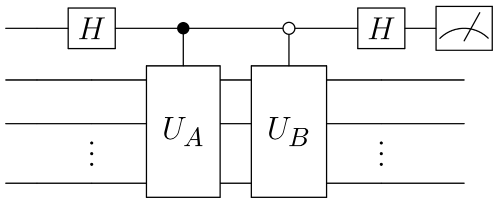


Figure 7.5: The interference circuit computes the overlap between k-qubit states, $|A\rangle$ and $|B\rangle$, with just $k + 1$ qubits.

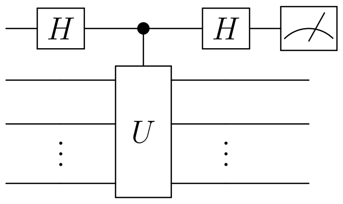


Figure 7.6: Circuit for the Hadamard Test.

7.4.2 Hadamard Test

The SWAP Test is a specific case of a more general procedure called the Hadamard Test. The Hadamard Test has a very simple and familiar form shown in Figure 7.6. This is essentially just the Controlled-U operation we focused on in Section 7.3. Moreover, the SWAP Test is just the case where $U =$ SWAP. Selecting other U makes the Hadamard Test give rise to a wide variety of applications. We list our benchmarked applications in Table 7.3. There are numerous additional applications of the Hadamard Test, such as training Quantum Boltzmann Machines [316], gradient evaluation [317, 318, 319, 320], and Jones polynomial approximation [321].

Application	Description
Variational Quantum Linear Solver [322, 323, 324]	Algorithm for solver large linear systems using NISQ hardware
Matrix elements of group representation [325, 326]	Group theory problem; U is essentially the Quantum Fourier Transform
Entanglement spectroscopy [327]	Computation of entanglement spectrum of arbitrary quantum states
Controlled Density Matrix Exponentiation (DME) [328]	Several appliations, e.g. for private quantum software [329]

Table 7.3: Applications of the Hadamard Test. Each corresponds to a different choice of Controlled-U.

7.4.3 Quantum Memory Architectures

Next, we investigate the use of fan-out to improve the implementation of quantum memory, which speeds up or enables many quantum algorithms [330, 331]. The high-level function of a quantum memory is similar to that of a classical memory: n index bits enumerate over 2^n memory cells. Following the notation of [332], we denote the n index bits as the $|b\rangle$ register and the 2^n memory cells as the $|m\rangle$ register. As in the classical case, we expect that setting the index register to $|i\rangle$ should allow us to retrieve the ith memory cell, $|m_i\rangle$. However, for a quantum memory, we also require the retrieval to work over superpositions of index qubits. For example, setting $|b\rangle$ to $\frac{1}{\sqrt{2}}[|000\rangle + |111\rangle]$ should retrieve the superposition, $\frac{1}{\sqrt{2}}[|m_0\rangle + |m_7\rangle]$.

In this section, we apply the fan-out primitive to both explicit and implicit quantum memories, which we define below. We demonstrate considerable improvements—exponential and linear respectively—over prior work, as summarized in Table 7.2. These improvements are important because the cost of quantum memory is often the principal bottleneck for realizing practical speedups. While it remains unclear if quantum memory architectures will be feasible [333, 305, 23, 16] even for future fault-tolerant devices, our proposed improvements at least justify a re-assessment of the feasibility.

Explicit Quantum Memory

In an explicit quantum memory, the 2^n memory cells are each explicitly stored in qubit registers. In this sense, an explicit quantum memory is akin to a 2^n–to–1 multiplexer or data selector from classical electronics. As discussed, the quantum variant should extend to the case where select lines are in superposition. Moreover, each of the 2^n memory cells is stored in a qubit register, so each memory cell can itself contain a quantum (superposition) state.

The dominant architecture for this explicit quantum memory is termed Quantum Random Access Memory. The bucket brigade design of QRAM was introduced in [334, 335] and cast

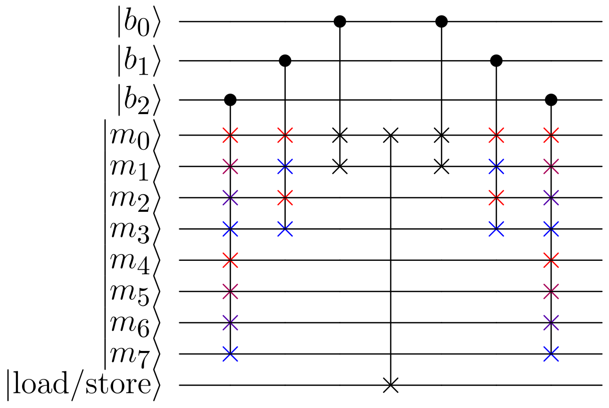


Figure 7.7: Architecture for an explicit quantum memory with $n = 3$ index qubits and $2^n = 8$ memory cells of bitwidth $W = 1$.

to the quantum circuit model in [305]. This bucket brigade QRAM requires $\sim 2 \cdot 2^n$ qubits and $O(W2^n)$ depth. Later work [306] was able to parallelize execution to achieve $O(Wn)$ depth, but requires an additional $\sim 6 \cdot 2^n$ ancilla qubits. We now present a novel architecture for explicit quantum memory that requires only $O(n)$ depth, with 0 ancilla qubits.

Figure 7.7 shows our architecture for $n = 3$ index qubits. There are $2^3 = 8$ explicit memory cells, each of single-qubit bitwidth $W = 1$. At a high level, the circuit performs a "migration" of the target memory cell into $|m_0\rangle$. Consider for example $|\vec{b} = 101\rangle$, which should access $|m_5\rangle$. The control on the MSB performs a SWAP between $|m_{7654}\rangle$ and $|m_{3210}\rangle$, moving $|m_5\rangle$ into $|m_1\rangle$. The control on the middle index does not activate, but the control on the LSB is activated and SWAPs $|m_1\rangle$ into the $|m_0\rangle$ destination. Finally, this qubit is swapped into the $|\text{load/store}\rangle$ register. The right half of the circuit reverses the earlier migrations, restoring the other memory cells to their original locations.

The efficiency of this architecture is enabled by the simultaneous execution of controlled SWAPs, which in turn is enabled by the fan-out primitive. As a result, the circuit depth is only $O(n)$. Moreover, while our example shows the $W = 1$ bitwidth case, it is apparent that with simultaneous fan-out, W is irrelevant to depth. By contrast, serialization would impose an additional linearity in W.

During the preparation of this chapter, another proposal was published for $O(n)$-depth and ancilla-free explicit quantum memory [336], which matches our asymptotic costs.

Implicit Quantum Memory

Next we consider implicit quantum memory. In this model, the 2^n memory cells represent classical (non-superposition) data that is known in advance. In such a case, there is no need to waste qubits to represent the classically-known memory cells. Instead, the memory can be stored implicitly through the classical control, a memory architecture that has been referred to as Quantum Read Only Memory [307].

Figure 7.8 shows an example implicit memory storing the first four prime numbers: $\{00 \to 2, 01 \to 3, 10 \to 5, 11 \to 7\}$. The resulting circuit has a simple form, enumerating all 2^n indices and associating each index with a corresponding pattern of $\oplus$ gates. Without fan-out, implicit memory has $O(W2^n)$ depth via the unary iteration optimization in [307]. However, simultaneous fan-out obviates the scaling in W. This is appealing, because for datasets such as images, the bitwidth (W) of each record exceeds the number of records.

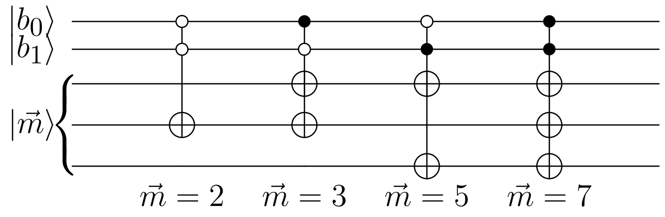


Figure 7.8: Implicit memory storing the first four prime numbers. The $W = 3$ bitwidth memory is implicitly defined through classical control, based on the pattern of $\oplus$'s. For anticipated applications, W can be large.

7.5 Technology Modeling: Trapped Ion

In this section, we model the implementation of fan-out on trapped ion quantum computers. Trapped ions feature long qubit coherence times [337] and gate fidelities exceeding 99.99%

and 99.9% for single- and two- qubit gates on current hardware [338, 339]. Furthermore, all N qubits can be simultaneously entangled via a global interaction known as the Global Mølmer–Sørensen (GMS) gate [340, 341]. Recent work [342, 227] has explicitly demonstrated how GMS is essentially equivalent to simultaneous fan-out. Moreover, in the past year, experimental work has merged demonstrating pulse shaping for global interactions [295, 343, 303] to support the use of GMS both for fan-out and for parallel two-qubit gates on disjoint qubits. Our focus here is on studying differences in speed and fidelity between simultaneous fan-out versus $N-1$ serialized CNOTs. For brevity and to maintain a focus on architectural themes, we omit many physical implementation details here.

Regarding the potential speedup, [342, 344, 303] assert that simultaneous fan-out via GMS is indeed linearly faster than serialized CNOTs. To evaluate the fidelity impact, we performed numerical simulations of fan-out via GMS for $N = 2$ to $N = 8$ qubits. We constructed a realistic error model that accounts for two sources of noise: overrotation and laser dephasing. Overrotation occurs due to the fact that the angle θ of the Mølmer-Sørensen rotation is sensitive to motional frequency drifts, and it has higher-order dependence on the motional states [345, 346, 347]. An overrotation error can be modeled by replacing θ by $(1+\epsilon)\theta$, where ϵ denotes the overrotation rate. Laser dephasing results from fluctuations of the optical path length [348, 346, 345].

For current trapped ion hardware, we conservatively estimate typical overrotation rates of 5%. We modeled GMS interaction times of 100 μs [342], contrasted against 80 ms laser coherence time [345]. To evaluate the sensitivity of our results to these parameters, we also modeled under three future scenarios: 5x lower overrotation rate, 5x longer laser coherence, and both improvements. Our simulations were performed using master-equation simulation in QuTiP [67]. We performed stochastic simulation over 100k runs per scenario. The fidelity results are shown in Figure 7.10.

Conceptually, overrotation errors affect simultaneous and serial equally. Meanwhile, laser dephasing affects serial more adversely, because the laser dephasing effect on the control qubit accumulates over the additional time required for $N-1$ consecutive CNOTs. Although simultaneous always outperforms serial on our simulations, the exact fidelity advantage is dependent on the parameter settings. For current technology (●), simultaneous has an almost 1% higher fidelity for $N = 8$. For the scenario with 5x longer laser coherence (▼), simultaneous has almost no fidelity advantage over serial. For the scenario with 5x lower overrotation (■), simultaneous again has a nearly 1% fidelity advantage over serial. Also, across all scenarios, the advantage of simultaneous fan-out increases for larger N, which is encouraging. The simulation results roughly agree with experimental work. For example [295] observed 93.4% fidelity inclusive of State Preparation and Measurement (SPAM) errors for a 4-qubit fan-out executed on hardware over a year ago, and Figure 7.10 suggests SPAM-exclusive fidelity of 99.3% on current hardware. As cloud access to trapped ion hardware emerges over the coming year, it will be possible to experimentally validate these simulations.

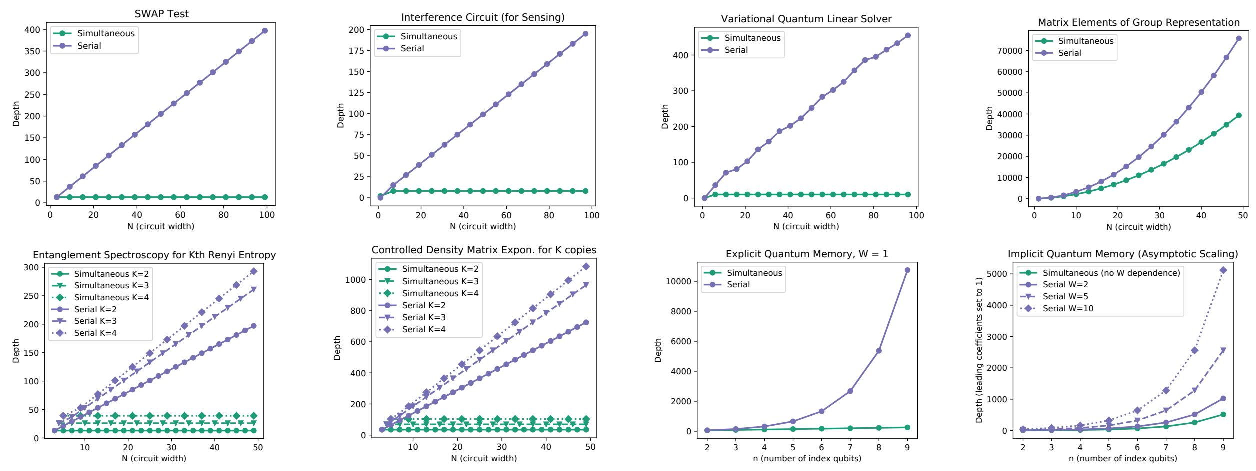


Figure 7.9: Depth (lower is better) for SWAP Test, Hadamard Test, and memory architecture benchmarks. We compare circuits compiled with our Controlled-U circuit synthesis procedure (which uses simultaneous fan-out) versus circuits that serialize the CNOTs.

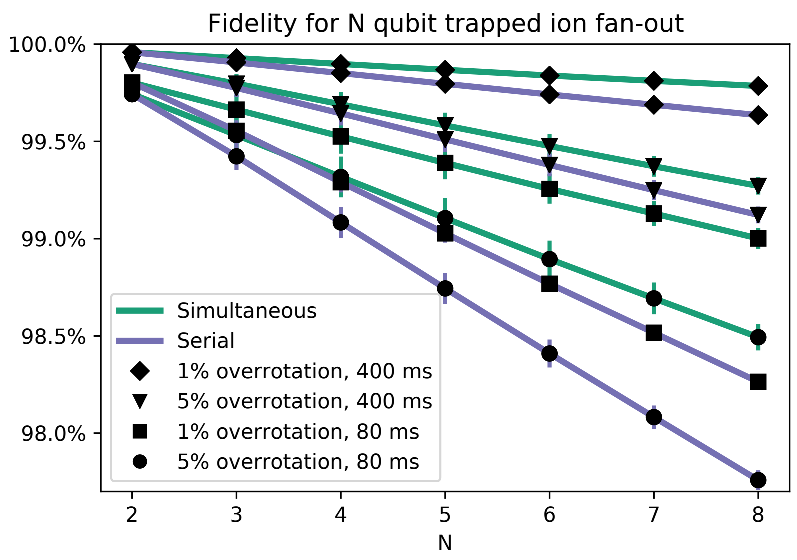


Figure 7.10: Simulation results for fan-out on trapped ion hardware. Sensitivity analysis performed under four {overrotation rate, laser coherence time} scenarios. For each scenario, we simulated fidelity for simultaneous versus serial. Results averaged across 100k stochastic runs per scenario, executed with 50k CPU-core hours on a large computing cluster.

7.6 Results

7.6.1 *Methodology*

We evaluated the exact depth reduction for eight applications: SWAP Tests (both traditional and interference circuit), Hadamard Tests (all four applications in Table 7.3), and memory architectures (both explicit and implicit). We compiled each benchmark, across a wide range of circuit widths, using both our fan-out based approach (Simultaneous) and the standard serialized approach with no fan-out (Serial). The results are plotted in Figure 7.9.

We also evaluated the fidelity advantage of simultaneous fan-out for the five most NISQ-friendly benchmarks. For each benchmark type, we found the largest circuit instance with fan-out of at most 8 qubits, matching the largest fan-out we simulated in Figure 7.10. Then, we estimated fidelity with a coarse metric: for each circuit, we assigned each gate a fidelity based on the current hardware "5% overrotation, 80 ms laser coherence" simulation in Figure 7.10. Multiplying together these gate fidelities gives an approximation for the total circuit fidelity (i.e. 1 - infidelity). We also performed this multiplication under the "1% overrotation, 80 ms laser coherence" future scenario with 5x lower overrotation. While these estimates are less accurate than full density matrix simulation, as we performed in Figure 7.10, they are informative from an Amdahl's Law perspective. In particular, single- and two- qubit gates are equally penalized in the Simultaneous and Serial circuits, so the Simultaneous circuits can only perform better when there are large fan-out gates.

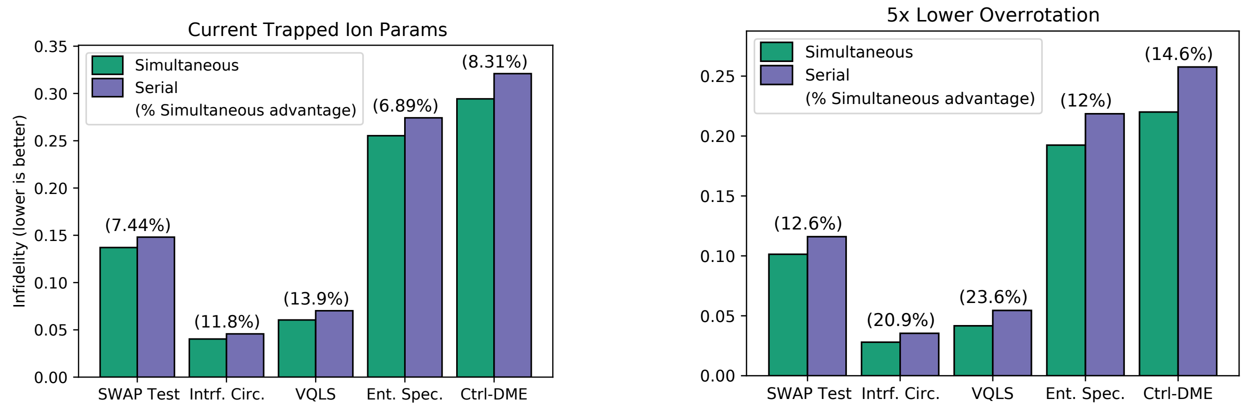


Figure 7.11: Infidelity estimates for five benchmarks.

7.6.2 Discussion

As mentioned in Section 7.5, simultaneous fan-out does genuinely give a linear speedup over serialization. Therefore, the depth reductions in Figure 7.9 translate directly to faster time-to-solution. For four of the eight benchmarked applications, the underlying U has constant depth, so our Simultaneous circuit also has constant depth. For the other four benchmarks, the underlying U has $\Omega(N)$ depth, so both Simultaneous and Serial have increasing depth with N. However, Simultaneous' scaling is still lower than Serial's by a linear factor.

Among our benchmarks, Variational Quantum Linear Solver and Controlled Density Matrix Exponentiation have particularly high fidelity advantage via Simultaneous Fan-out. Our results also demonstrate the sensitivity to the underlying trapped ion hardware's error parameters. For example, VQLS has a 13.9% Serial$\rightarrow$Simultaneous infidelity reduction on current hardware versus a 20.9% reduction on future hardware with 5x lower overrotation.

On current hardware, fidelity is the primary system bottleneck. As such, the fidelity improvement of simultaneous fan-out justifies its use in NISQ machines. 7–24% reductions in infidelity on 8-qubit circuits are equivalent to months of hardware progress, but our optimization requires no new hardware. As a practical message to hardware providers, we emphasize that exposing global interactions to software will lead to substantial improvements in both fidelity and speed for NISQ applications.

7.7 Future Work: Superconducting Qubits

Global interaction can be realized for many technologies, but superconducting qubits—which are currently the frontrunner in commercial activity—are a notable exception. To the best of our knowledge, there are no prior implementations of fan-out on superconducting devices. In this section, we demonstrate that superconducting quantum computers can in fact perform simultaneous fan-out.

We first examine the implementation of a CNOT with superconducting qubits. The CNOT is not a natural physical interaction between qubits. Instead, it is performed through a sequence of more primitive physical interactions between qubits. On Google and Rigetti superconducting quantum hardware, CNOT can be realized by a sequence of iSWAP interactions, which are similar to ordinary SWAPs. However, this seems incompatible with simultaneous fan-out, which conceptually requires concurrent reads on the control qubit. By contrast, iSWAP performs both reads and writes on the control qubit since its state is swapped with the target.

An alternative two-qubit interaction called Cross-Resonance [233, 234] is better suited. The Cross-Resonance interaction is used to perform CNOT gates on IBM's devices. It has less restrictive hardware requirements than iSWAP, so Cross-Resonance could be performed on Google and Rigetti hardware as well. Critically, the Cross-Resonance interaction only reads the control qubit, so it does not suffer the immediate barrier to fan-out that iSWAP does.

Although the control qubit *state* is unaffected during Cross-Resonance, the interaction requires (somewhat counterintuitively) driving the control qubit with a microwave pulse. However, by setting this microwave pulse to the frequency of the target qubit, the target qubit will rotate conditioned on the state of the control qubit. This physical interaction easily converts to CNOT through a single-qubit postprocessing gate.

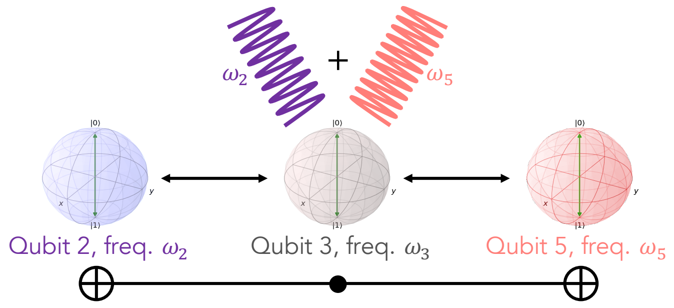

ω_2
+
ω_5
$|0\rangle$
$|1\rangle$
Qubit 2, freq. ω_2
Qubit 3, freq. ω_3
Qubit 5, freq. ω_5

Figure 7.12: Schematic of fan-out using Cross-Resonance on superconducting qubits. The control qubit (3) is driven with the sum of waves at the targets' frequencies, ω_2 and ω_5.

Figure 7.12 illustrates how we can extend this Cross-Resonance interaction to engineer fan-out. In this example, qubit 3 is the control and qubits 2 and 5 are the two targets. To perform the CNOT from 3 to 2 (5), we would drive qubit 3 with microwave at frequency ω_2 (ω_5). However, if we instead drive qubit 3 with the *summation* of two sine waves at frequencies ω_2 and ω_5, then we effectively perform both CNOTs simultaneously. The resulting pulse sequence has a linear speedup over serialization, as desired.

We experimentally realized this specific example of fan-out from qubit 3 to qubits 2 and 5 using IBM's Paris quantum computer. We performed our experiment using OpenPulse [100, 207, 21], an interface that enables low-level access of quantum computers through Arbitary Waveform Generators. This level of access is required since we need to drive qubit 3 with an unconventional sum-of-waves pulse. We also use a technique called sideband modulation, which is needed since the qubit 3 drive is configured to oscillate at ω_3 by default.

Moreover, in practice, high fidelity Cross-Resonance interactions require an echo sequence [256] and active cancellation pulses on the target qubits [255, 257]. Additionally, we had to calibrate a phase offset for the sideband to compensate for accumulated phase on the coaxial cable transitioning from room temperature electronics to the fridge [207, 349].

Figure 7.13 shows our experimental results. We generated the GHZ state, $\frac{|000\rangle+|111\rangle}{\sqrt{2}}$, by performing a NOT gate on qubit 3 and then fanning out its state to qubits 2 and 5. Ideally, this would result in $|000\rangle$ and $|111\rangle$ each with 50% probability. With simultaneous fan-out, we achieved 31% and 29% respectively. Serialization achieved 42% and 36% respectively.

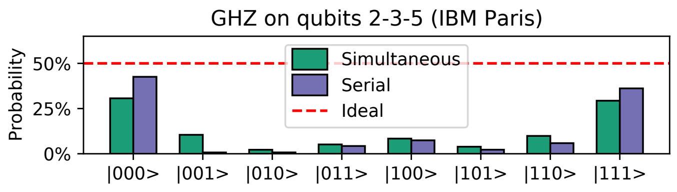


Figure 7.13: OpenPulse results from 8000 × 2 repetitions on IBM Q Paris. The ideal output is 50% $|000\rangle$ and 50% $|111\rangle$.

While the GHZ state produced with serial fan-out is better than the one produced with simultaneous fan-out, we emphasize that the simultaneous version ran almost twice as fast. Most importantly, our experiment affirms that simultaneous fan-out is possible at all on superconducting qubits. Section 7.10 has further discussion regarding the scalability of simultaneous fan-out on superconducting qubits.

7.8 Conclusion

At a high level, this work validates the importance of hardware-software codesign. Our core result is driven from the hardware → software observation that the exclusive activation structural hazard is not necessary in quantum computing. By exploiting simultaneous fan-out, we are able to synthesized optimized circuit schedules for Controlled-U, which is important in NISQ workloads. In the software → hardware direction, our results suggest a number of priorities for future hardware development—in particular, the importance of exposing global interactions. Moreover, our demonstration of simultaneous fan-out in superconducting qubits they could be brought to parity with trapped ions.

In current systems, our results affirm a linear speedup from fan-out. In the NISQ era, algorithms will require millions of iterations [17], so quantum execution speedups translate to direct reductions in time-to-solution. This opportunity is particularly pronounced on trapped ions, which operate at relatively slow kHz speeds. In addition to the circuit execution speedup, our simulations show 7–24% infidelity reductions from simultaneous fan-out. This is validated by our trapped ion simulation with a realistic noise model. Our experimental results from superconducting qubits are also promising, though our emphasis is on the mere fact that simultaneous fan-out is possible at all on superconducting qubits.

A number of interesting future directions arise from this work. On the hardware side, we propose experimental realization of our circuits on larger machines. Also, global interactions via 'Rydberg gates' are natural on neutral atoms [350], which are emerging as another major qubit technology. Our results will extend naturally to neutral atoms, as well as other qubit technologies. On the software side, we propose further investigation of compilation in view of global interactions. [303] suggests that global interactions could in fact give an $O(N^2)$ speedup, whereas we only explore linear speedups in this work. Finding such quadratic speedups could further accelerate the realization of practical quantum computing.

7.9 Background on Quantum Computing

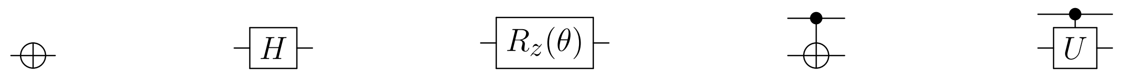
To keep this chapter self-contained, we provide necessary background on quantum computing in this Section. To maintain accessibility, we emphasize the circuit model of quantum computing, rather than its linear algebraic formulation.

7.9.1 Qubits

A qubit (quantum bit) is defined by two states, denoted $|0\rangle$ and $|1\rangle$. Just as classical bits can be implemented by a variety of underlying physical representations like magnetization in disks or capacitor charge in RAM, qubits can be fabricated from a variety of underlying quantum technologies. This includes discrete charge levels in superconducting qubits or motional modes in trapped ion qubits.

The state of a qubit can be written as the linear combination $a\,|0\rangle + b\,|1\rangle$, subject to a normalization condition, $|a|^2 + |b|^2 = 1$. This is richer state space that can be captured by a classical bit, which is either $|0\rangle$ or $|1\rangle$. For example, $\frac{1}{\sqrt{2}}[|0\rangle + |1\rangle]$ is a *superposition* qubit state: the qubit has equal components in $|0\rangle$ and $|1\rangle$.

The state of a qubit can be changed by a gate. Figure 7.14 depicts the gates we use in this chapter. Each gate has an input wire(s) and an output wire(s). The first gate is just the classical NOT gate, which interchanges $|0\rangle$ and $|1\rangle$. The next gate is the Hadamard gate, which is an intrinsically quantum gate that creates superposition. For example, applying H to a $|0\rangle$ creates the equal superposition $\frac{1}{\sqrt{2}}[|0\rangle + |1\rangle]$. The $R_z(\theta)$ gate is another quantum gate which applies a phase. Phase is helpfully visualized as a θ displacement in longitude on a sphere; however, for our purposes its underlying meaning is unimportant. Gates (d) and (e) act on pairs of qubits (wires). The CNOT is a Controlled-NOT, which applies a NOT to the bottom qubit, iff the top qubit is $|1\rangle$. The top qubit itself is unaffected. The Controlled-U is simply a generalization, where U represents some single qubit gate that is only activated if the top qubit is $|1\rangle$.



(a) NOT (b) Hadamard (c) R_z rotation (d) CNOT (e) Controlled-U

Figure 7.14: Gates used in this chapter.

While a qubit can carry a rich state space, it snaps to either $|0\rangle$ or $|1\rangle$ upon measurement. This process is fundamentally stochastic: the probability of measuring $|0\rangle$ is $|a|^2$ and $|1\rangle$ is $|b|^2$, which justifies the normalization condition. For example, under the $\frac{1}{\sqrt{2}}[|0\rangle + |1\rangle]$ state, the probability of measuring $|0\rangle$ is $(\frac{1}{\sqrt{2}})^2 = \frac{1}{2}$ which is indeed an equal superposition. The measurement operation is visually denoted as $\meter$, which terminates a wire.

7.9.2 Quantum circuits

Quantum programs are expressed as quantum circuits which, like Boolean circuits, carry wires representing qubits through a sequence of gates. An example quantum circuit is shown below in Figure 7.15. It can be read as a timeline from left to right. As indicated, the *width* is the number of qubits the circuit acts on. In addition to data qubits that encode input/output, quantum circuits often use extra *ancilla* qubits that store temporary results. The *depth* is the length of the critical path. Thus, width and depth respectively capture the space and time costs of a quantum circuit.

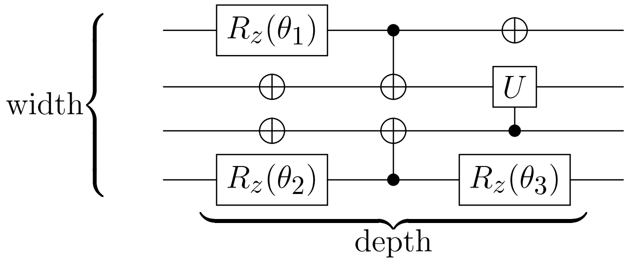


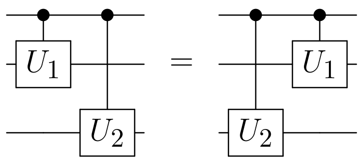
Figure 7.15: An example quantum circuit with a width of 4 qubits and a depth of 3 layers.

Figure 7.15 exemplifies data parallelism—no qubit is ever idle. Such speedups are especially important in quantum computing because qubits generally have short coherence windows for useful computation.

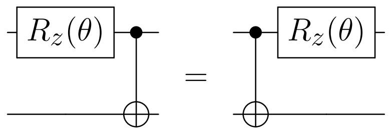
7.9.3 Commutativity

Every quantum circuit has an underlying program dependency graph that enforces the execution order of gates. Naively, one can construct a program dependency graph that simply adds forward dependencies from each quantum gate to subsequent quantum gates in the circuit-timeline. However, this dependency graph can often be relaxed due to commutativity, where two quantum gates can be applied in either order.

Many commutativity relationships exist between gates. For our work, we only rely on the two relationships depicted in Figure 7.16. The left equivalence shows that two Controlled-U_i gates commute when they have different targets. This is clear because controlled gates leave the control qubit unchanged, so their order is unimportant. The right equivalence shows that R_z-type gates commute with controls of controlled gates (such as CNOTs). This relationship has no classical analogue, but the underlying intuition is that R_z gates don't affect the $|0\rangle$ vs. $|1\rangle$ balance of a qubit, so their order relative to a control is unimportant. Both of these commutativity rules are used in our Controlled-U circuit synthesis procedure in Section 7.3.



(a) Different targets.



(b) R_z gates commute with controls.

Figure 7.16: Two commutativity rules encountered in this chapter.

7.10 Scalability of Simultaneous Fan-out on Superconducting Qubits

Section 7.7 presented our experimental realization of simultaneous fan-out on superconducting qubits with OpenPulse on IBM Q Paris. With simultaneous fan-out, we produced a three qubit GHZ state (50% $|000\rangle$ and 50% $|111\rangle$) with 31% and 29% measurement outcomes for $|000\rangle$ and $|111\rangle$ respectively. Serialized fan-out on the same machine achieved better outcomes of 42% and 36% outcomes, required almost twice the runtime. Although simultaneous fan-out had lower fidelity, the speedup is encouraging, because superconducting qubits have short coherence lifetimes, so faster operations lead to significant fidelity improvements [280, §II. E.]. Moreover, when we consider larger width circuits, faster fan-out on a subset of qubits can improve the quality of the other qubits which decohere for less time. Finally, anticipated increases to the sampling rate of Arbitrary Waveform Generators should improve the fidelity of the simultaneous fan-out operation. Recent papers have also proposed different techniques that could be used to realize many-body interactions in superconducting systems [301, 351, 352, 353], but our work is the first experimental proof-of-concept. Our work can be viewed a way to engineer crosstalk (unwanted interference between neighboring qubits) for good. However, key questions remaing in scaling up simultaneous fan-out on superconducting hardware.

An immediate barrier to scaling our procedure to more target qubits is that each control-target pair must be connected in hardware. On superconducting qubit platforms, connectivity is typically sparse. For example, on IBM Q Paris's device topology, the maximum degree is 3, and most qubits are connected to just one or two neighbors. Scaling the connectivity will be a challenge. However, we note that fan-out does not require all-to-all connectivity. Instead, we require a star topology, where a single (control) qubit is connected to every other qubit. Such star topologies have been realized experimentally with 10 qubits connected to a single bus [354]. Moreover, star topology is also useful for Hamiltonian simulation circuits [355], so there are numerous other quantum subroutines that would also benefit.

A second consideration is that summing waves for each target qubit's frequency (as in Figure 7.12) will not scale since the maximum amplitude of Arbitrary Waveform Generators is power-constrained. We propose two possible solutions to this. On frequency tunable devices (where ω_q for each qubit can be controlled), we can simply tune all target qubits to a common frequency during fan-out. Then, the control qubit can be driven at this single common frequency, bypassing the summation of multiple waves. The other solution pertains to fixed-frequency devices. Here, we propose that rectangular-topology qubits could be fabricated with frequencies according to a checkerboard pattern. In such an arrangement, just two colors (frequencies) are needed to ensure no frequency collisions between neighboring qubits. During fan-out, the control qubit can be driven at the sum of just two frequencies, averting the scalability issue.

While these proposed solutions are sound in theory, practical realization will be challenging due to experimental nuances. For example, current qubit fabrication technologies are imprecise and stochastic [356], so fabricating qubit frequencies in a checkerboard pattern will be difficult. Thus, more experimental progress will be needed to scale fan-out on superconducting hardware. These hardware-software codesign considerations are valuable in closing the gap from NISQ hardware to practical applications. We propose further work to evaluate simultaneous fan-out with superconducting qubits.

CHAPTER 8
CONCLUSION

Figure 8.1 summarizes the six papers presented in Chapters 2–7 in terms of the layers of the stack that they each address. Emphatically, the optimizations cut across these layers, validating the overarching thesis statement. Under the standard paradigm of a stack with rigid abstraction barriers between layers, it is not possible to achieve these optimizations.

	Qutrits	Partial Compilation	Simultaneous Measurement	O(N^3) VQE	OpenPulse	Quantum Fan-out
Application			✓	✓		
Algorithm	✓	✓	✓	✓	✓	✓
Compiler	✓	✓	✓	✓	✓	✓
Control		✓			✓	✓
Hardware	✓	✓			✓	✓

Figure 8.1: Each of the six papers presented in this dissertation introduced an optimization that cuts across layers spanning the quantum stack. The columns correspond to Chapter 2 (Qutrits) through Chapter 7 (Quantum Fan-out).

We propose two thematic areas for exciting future work. First, we call for more research extending into the application layer. From the six papers presented here, Chapters 4 and 5 addressed this layer and they did so with particularly promising results. We suggest that additional gains can be realized by tailoring quantum computation to actual end-user applications. By contrast, most existing work stops at the algorithm layer. In our own ongoing research, we are studying the application of quantum computation to practical problems in energy and finance industries, where we expect application layer optimizations to manifest. In addition, further exploration at the boundary between the application and algorithm layer could be fruitful. For example, a recently proposed quantum algorithm for Maximum

Independent Set (MIS) [357] may have broader implications than MIS by retargeting it to applications involving constrained optimization.

We also propose further work on the theme of pulse-level error mitigation. While full error *correction* is outside the scope of the NISQ era, there are a number of promising approaches for error *mitigation*. Error mitigation can suppress effective noise significantly below what would be expected from isolated gate error rates. Moreover, with the advent of software interfaces for pulse-level control, new techniques have emerged for analog error mitigation strategies that have more flexibiltiy than gate-based techniques. These new techniques include zero-noise extrapolation [358] and echoed error cancellation [255, 230]. These error mitigation techniques reveal new opportunities for the compiler to perform optimization not only at the logical noise-unaware level, but also in the physical noise-aware setting.

Although it is still in its early days, the field of quantum computing bears many exciting possibilities ahead. It is exactly because of these exciting possibilities—in areas like molecular simulation and logistics optimization—that it is doubly important for us to make quantum computing a practical reality soon. The optimizations presented in this paper suggest that we can accelerate the timeline for quantum computing by breaking from the conventional wisdom of stack design. By instead designing a stack that encourages optimizations that cross between layers, we can achieve near-term efficiency gains that are otherwise invisible.

APPENDIX A

CURRICULUM VITAE

Pranav Gokhale

Website: pranavgokhale.com
Email: pranav@super.tech
LinkedIn: pgokhale
GitHub: github.com/singular-value

Education

University of Chicago Chicago, IL
Ph.D. in Computer Science, Advisor: Frederic T. Chong 2017–2020
- Thesis: "Full-Stack, Cross-Layer Optimizations for Quantum Computing"
- Awarded NDSEG (DoD) and Eckhardt (UChicago PSD) fellowships

University of Chicago Chicago, IL
M.S. in Computer Science 2017–2019
- Thesis: "Asymptotic Improvements to Quantum Circuits via Qutrits"

Princeton University Princeton, NJ
B.S.E. in Computer Science with Certificate in Engineering Physics; GPA: 3.75 2011–2015
- Graduated magna cum laude. Inducted into Tau Beta Pi and Sigma Xi
- Conducted research for two years with Professors Mung Chiang and Margaret Martonosi
- Took graduate-level coursework in quantum computation

University of Waterloo Waterloo, ON (Canada)
Undergraduate School in Experimental Quantum Information Processing Summer 2014

Publications

[1] J. M. Baker, C. Duckering, **P. Gokhale**, N. C. Brown, K. R. Brown, and F. T. Chong, "Improved quantum circuits via intermediate qutrits", *ACM Transactions on Quantum Computing*, 2020.

[2] Y. Ding, **P. Gokhale**, S. F. Lin, R. Rines, T. Propson, and F. T. Chong, "Systematic crosstalk mitigation for superconducting qubits via frequency-aware compilation", in *Proceedings of the 53nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2020.

[3] R. Eskandarpour, **P. Gokhale**, A. Khodaei, F. T. Chong, A. Passo, and S. Bahramirad, "Quantum computing for enhancing grid security", *IEEE Transactions on Power Systems*, vol. 35, no. 5, pp. 4135–4137, 2020.

[4] **P. Gokhale**, O. Angiuli, Y. Ding, K. Gui, T. Tomesh, M. Suchara, M. Martonosi, and F. Chong, "Minimizing state preparations in variational quantum eigensolver by partitioning into commuting families", in *Proceedings of the IEEE International Conference on Quantum Computing and Engineering (QCE2020)*, 2020.

[5] **P. Gokhale**, J. M. Baker, C. Duckering, F. Chong, N. Brown, and K. Brown, "Extending the frontier of quantum computers with qutrits", *IEEE Micro*, vol. 40, pp. 64–72, 2020.

[6] **P. Gokhale**, A. Javadi-Abhari, N. Earnest, Y. Shi, and F. Chong, "Optimized quantum compilation for near-term algorithms with openpulse", in *Proceedings of the 53nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2020.

[7] **P. Gokhale**, S. Koretsky, S. Huang, S. Majumder, A. Drucker, K. Brown, and F. Chong, "Quantum fan-out: Circuit optimizations and technology modeling", *arXiv Preprint*, vol. abs/2007.04246, 2020.

[8] K. Gui, T. Tomesh, **P. Gokhale**, Y. Shi, F. Chong, M. Martonosi, and M. Suchara, "Term grouping and travelling salesperson for digital quantum simulation", *arXiv Preprint*, 2020.

[9] J. McClean, K. J. Sung, I. Kivlichan, Y. Cao, C. Dai, E. Fried, C. Gidney, B. Gimby, **P. Gokhale**, T. Haner, T. Hardikar, V. Havl'ivcek, O. Higgott, C. Huang, J. Izaac, Z. Jiang, X. Liu, S. McArdle, M. Neeley, T. O'Brien, B. O'Gorman, I. Ozfidan, M. D. Radin, J. Romero, N. Rubin, N. P. D. Sawaya, K. Setia, S. Sim, D. S. Steiger, M. Steudtner, Q. Sun, W. Sun, D. Wang, F. Zhang, and R. Babbush, "Openfermion: The electronic structure package for quantum computers", *Quantum Science and Technology*, 2020.

[10] Y. Shi, **P. Gokhale**, P. Murali, J. M. Baker, C. Duckering, Y. Ding, N. Brown, C. Chamberland, A. Javadi-Abhari, A. Cross, D. Schuster, K. Brown, M. Martonosi, and F. Chong, "Resource-efficient quantum computing by breaking abstractions", *Proceedings of the IEEE*, vol. 108, pp. 1353–1370, 2020.

[11] T. Tomesh, **P. Gokhale**, E. R. Anschuetz, and F. Chong, "Coreset clustering on small quantum computers", *IEEE Quantum Week AQAI (Applied Quantum Artificial Intelligence) Workshop*, vol. arXiv preprint: abs/2004.14970, 2020.

[12] **P. Gokhale**, J. M. Baker, C. Duckering, N. Brown, K. Brown, and F. Chong, "Asymptotic improvements to quantum circuits via qutrits", *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*, pp. 554–566, 2019.

[13] **P. Gokhale** and F. Chong, "O(n^3) measurement cost for variational quantum eigensolver on molecular hamiltonians", *arXiv Preprint*, 2019.

[14] **P. Gokhale**, Y. Ding, T. Propson, C. Winkler, N. Leung, Y. Shi, D. I. Schuster, H. Hoffmann, and F. 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*, 2019, pp. 266–278.

[15] Y. Shi, N. Leung, **P. Gokhale**, Z. Rossi, D. Schuster, H. Hoffmann, and F. Chong, "Optimized compilation of aggregated instructions for realistic quantum computers", *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019.

[16] A. JavadiAbhari, **P. Gokhale**, A. Holmes, D. Franklin, K. Brown, M. Martonosi, and F. Chong, "Optimized surface code communication in superconducting quantum computers", *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 692–705, 2017.

[17] **P. Gokhale**, "Implementation of square root function using quantum circuits", in *The Global Undergraduate Awards*, 2014.

[18] E. Silver, J. D. Gillaspy, **P. Gokhale**, E. Kanter, N. Brickhouse, R. Dunford, K. Kirby, T. Lin, J. W. Mcdonald, D. Schneider, S. Seifert, and L. Young, "Work towards experimental evidence of hard x-ray photoionization in highly charged krypton", in *AIP Conference Proceedings*, 2011.

Prior Experience

Telepath (Stealth Startup) — Atherton, CA
Product engineer and designer — Jul. 2017 – Dec. 2017
- Built technical and product infrastructure during extended internship prior to joining UChicago

Quora — Mountain View, CA
Software Engineer — Jul. 2015 – Jul. 2017
- Primary engineer for Quora's i18n efforts and Quora Sessions
- Mentored several fulltime engineers and interns
- Also interned at Quora in summer 2014

Princeton University — Princeton, NJ
Researcher and software engineer — 2013 – 2015
- Quantum architecture researcher with Martonosi group
  - Implemented a scheduler for entangled qubit pairs in Multi-SIMD architecture
  - Implemented recursive graph partitioner to optimize qubit placement for surface code
- Database researcher for Wikimedia through Facebook Open Academy
  - Worked on open-source project to improve scalability of Wikipedia's parser
- Systems researcher in Chiang lab (EDGE)
  - Built wearable hardware that detects finger gestures and uses machine learning to infer textual input
- Independent quantum researcher in reversible arithmetic circuit design, led to Global Undergraduate Award

Siemens — Mulheim, NRW (Germany)
Data Science Intern — Summer 2014
- Analyzed and optimized solutions to differential equations governing emissions from gas turbines

National Institute of Standards and Technology (NIST) — Gaithersburg, MD
Student Researcher — 2006 – 2010
- Researched x-ray photoionization of highly charged ions and published paper in CAARI 2010
- Optimized key distribution rates in quantum cryptography link

Scholarships and Awards

• Argonne National Laboratory CRI Innovator Post-Doctoral Fellowship	2020–current
• IEEE QCE Quantum Week Best Paper Award	2020
• IBM Q Global Best Paper Award	2020
• IEEE MICRO Top Pick Award	2020
• UChicago Graduate Student Leadership and Recognition Award	2019
• DoD National Defense Science and Engineering Graduate Fellowship (NDSEG)	2017–2020
• UChicago PSD Eckhardt Fellowship	2017–2020
• Numerous hackathon awards, e.g. 2018 Vatican City Hackathon & 2015 NYU Abu Dhabi Hackathon	2012 –2018
• NSF Honorable Mention for Graduate Research Fellowship Program (GRFP)	2017
• Magna cum laude, Princeton University	2015
• Inducted into Tau Beta Pi and Sigma Xi, Princeton University	2015
• Ruhr Fellowship (Bochum, Germany)	2014
• Highly Commended Research in Global Undergraduate Awards	2014
• National Advanced Placement (AP) Scholar	2011
• Maryland Distinguished Scholar	2011

Page 3 of 4

Service

• Volunteer for several outreach activities: Hour of Code, ACM-W mentorship program, compileHer	2017 –current
• Research mentor for six undergraduates including four underrepresented minorities	2017 –current
• Session Chair, IEEE Quantum Week (QCE) Conference	2020
• Program Committee, Applied Quantum Artificial Intelligence (AQAI) Workshop	2020
• Program Committee, Quantum Resource Estimation (QRE) Workshop	2020
• Organizer, Quantum + Power Grid Optimization Workshop	2020

REFERENCES

[1] Yao-Min Di and Hai-Rui Wei. Elementary gates for ternary quantum logic circuit. *arXiv preprint arXiv:1105.5485*, 2011.

[2] William M Kirby and Peter J Love. Contextuality in the variational quantum eigensolver. *arXiv preprint arXiv:1904.02260*, 2019.

[3] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A*, 52:3457–3467, Nov 1995. doi: 10.1103/PhysRevA.52.3457. URL `https://link.aps.org/doi/10.1103/PhysRevA.52.3457`.

[4] Craig Gidney. Constructing large controlled nots. http://algassert.com/circuits/2015/06/05/Constructing-Large-Controlled-Nots.html, 2015.

[5] Quantum devices and simulators. `www.research.ibm.com/ibm-q/technology/devices/`, 2018.

[6] 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.

[7] Gavin E. Crooks. Performance of the quantum approximate optimization algorithm on the maximum cut problem, 2018.

[8] Zsolt Baranyai. On the factorization of the complete uniform hypergraphs. *Infinite and finite sets*, 1974.

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

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

[11] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. *Journal of statistical physics*, 22(5):563–591, 1980.

[12] Richard P Feynman. Simulating physics with computers. *Int. J. Theor. Phys*, 21(6/7), 1982.

[13] Jens Koch, M Yu Terri, Jay Gambetta, Andrew A Houck, DI Schuster, J Majer, Alexandre Blais, Michel H Devoret, Steven M Girvin, and Robert J Schoelkopf. Charge-insensitive qubit design derived from the cooper pair box. *Physical Review A*, 76(4): 042319, 2007.

[14] F. Arute, K. Arya, R. Babbush, D. Bacon, Joseph C. Bardin, R. Barends, R. Biswas, S. Boixo, Fernando G. S. L. Brandão, David A. Buell, Brian J. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, M. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, Mike Lindmark, E. Lucero, D. Lyakh, Salvatore Mandrà, J. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, John C. Platt, C. Quintana, E. Rieffel, P. Roushan, N. Rubin, D. Sank, K. Satzinger, V. Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, A. Vainsencher, Benjamin Villalonga, T. White, Z. Yao, P. Yeh, Adam Zalcman, H. Neven, and J. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574:505–510, 2019.

[15] Edwin Pednault, John A Gunnels, Giacomo Nannicini, Lior Horesh, and Robert Wisnieff. Leveraging secondary storage to simulate deep 54-qubit sycamore circuits. *arXiv preprint arXiv:1910.09534*, 2019.

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

[17] Pranav Gokhale, Jonathan M Baker, Casey Duckering, Natalie C Brown, Kenneth R Brown, and Frederic T Chong. Asymptotic improvements to quantum circuits via qutrits. In *Proceedings of the 46th International Symposium on Computer Architecture*, pages 554–566, 2019.

[18] 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. ACM, 2019.

[19] Pranav Gokhale, Olivia Angiuli, Yongshan Ding, Kaiwen Gui, Teague Tomesh, Martin Suchara, Margaret Martonosi, and Frederic T. Chong. Minimizing state preparations in variational quantum eigensolver by partitioning into commuting families, 2019.

[20] Pranav Gokhale and Frederic T Chong. $o(n^3)$ measurement cost for variational quantum eigensolver on molecular hamiltonians. *arXiv preprint arXiv:1908.11857*, 2019.

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

[22] Pranav Gokhale, Samantha Koretsky, Shilin Huang, Swarnadeep Majumder, Andrew Drucker, Kenneth R Brown, and Frederic T Chong. Quantum fan-out: Circuit optimizations and technology modeling. *arXiv preprint arXiv:2007.04246*, 2020.

[23] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[24] Ivan Kassal, James D Whitfield, Alejandro Perdomo-Ortiz, Man-Hong Yung, and Alán Aspuru-Guzik. Simulating chemistry using quantum computers. *Annual review of physical chemistry*, 62:185–207, 2011. doi: 10.1146/annurev-physchem-032210-103512.

[25] Yongshan Ding, Adam Holmes, Ali Javadi-Abhari, Diana Franklin, Margaret Martonosi, and Frederic Chong. Magic-state functional units: Mapping and scheduling multi-level distillation circuits for fault-tolerant quantum architectures. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 828–840. IEEE, 2018.

[26] Ali Javadi-Abhari, Pranav Gokhale, Adam Holmes, Diana Franklin, Kenneth R. Brown, Margaret Martonosi, and Frederic T. Chong. Optimized surface code communication in superconducting quantum computers. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17, pages 692–705, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4952-9. doi: 10.1145/3123939.3123949. URL `http://doi.acm.org/10.1145/3123939.3123949`.

[27] Gian Giacomo Guerreschi and Jongsoo Park. Two-step approach to scheduling quantum circuits. *Quantum Science and Technology*, 3(4):045003, 2018.

[28] Archimedes Pavlidis and Emmanuel Floratos. Arithmetic circuits for multilevel qudits based on quantum fourier transform. *arXiv preprint arXiv:1707.08834*, 2017.

[29] J. Randall, S. Weidt, E. D. Standing, K. Lake, S. C. Webster, D. F. Murgia, T. Navickas, K. Roth, and W. K. Hensinger. Efficient preparation and detection of microwave dressed-state qubits and qutrits with trapped ions. *Phys. Rev. A*, 91:012322, 01 2015. doi: 10.1103/PhysRevA.91.012322. URL `https://link.aps.org/doi/10.1103/PhysRevA.91.012322`.

[30] J. Randall, A. M. Lawrence, S. C. Webster, S. Weidt, N. V. Vitanov, and W. K. Hensinger. Generation of high-fidelity quantum control methods for multilevel systems. *Phys. Rev. A*, 98:043414, 10 2018. doi: 10.1103/PhysRevA.98.043414. URL `https://link.aps.org/doi/10.1103/PhysRevA.98.043414`.

[31] Cirq: A python framework for creating, editing, and invoking noisy intermediate scale quantum (NISQ) circuits. `https://github.com/quantumlib/Cirq`, 2018.

[32] Michael A Nielsen and Isaac L Chuang. Quantum computation and quantum information (10th anniv. version), 2010.

[33] Jean-Luc Brylinski and Ranee Brylinski. Universal quantum gates. In *Mathematics of quantum computation*, pages 117–134. Chapman and Hall/CRC, 2002.

[34] Ashok Muthukrishnan and C. R. Stroud. Multivalued logic gates for quantum computation. *Phys. Rev. A*, 62:052309, Oct 2000. doi: 10.1103/PhysRevA.62.052309. URL `https://link.aps.org/doi/10.1103/PhysRevA.62.052309`.

[35] A. B. Klimov, R. Guzmán, J. C. Retamal, and C. Saavedra. Qutrit quantum computer with trapped ions. *Phys. Rev. A*, 67:062313, Jun 2003. doi: 10.1103/PhysRevA.67.062313. URL `https://link.aps.org/doi/10.1103/PhysRevA.67.062313`.

[36] Michael Kues, Christian Reimer, Piotr Roztocki, Luis Romero Cortés, Stefania Sciara, Benjamin Wetzel, Yanbing Zhang, Alfonso Cino, Sai T. Chu, Brent E. Little, David J. Moss, Lucia Caspani, José Azaña, and Roberto Morandotti. On-chip generation of high-dimensional entangled quantum states and their coherent control. *Nature*, 546:622 EP –, 06 2017. URL `http://dx.doi.org/10.1038/nature22986`.

[37] T. Bækkegaard, L. B. Kristensen, N. J. S. Loft, C. K. Andersen, D. Petrosyan, and N. T. Zinner. Superconducting qutrit-qubit circuit: A toolbox for efficient quantum gates. *arXiv preprint arXiv:1802.04299*, 2018.

[38] A. Fedorov, L. Steffen, M. Baur, M. P. da Silva, and A. Wallraff. Implementation of a toffoli gate with superconducting circuits. *Nature*, 481:170 EP –, Dec 2011. URL `https://doi.org/10.1038/nature10713`.

[39] Andrew D. Greentree, S. G. Schirmer, F. Green, Lloyd C. L. Hollenberg, A. R. Hamilton, and R. G. Clark. Maximizing the hilbert space for a finite number of distinguishable quantum states. *Phys. Rev. Lett.*, 92:097901, Mar 2004. doi: 10.1103/PhysRevLett.92.097901. URL `https://link.aps.org/doi/10.1103/PhysRevLett.92.097901`.

[40] Mozammel H. A. Khan and Marek A. Perkowski. Quantum ternary parallel adder/subtractor with partially-look-ahead carry. *J. Syst. Archit.*, 53(7):453–464, July 2007. ISSN 1383-7621. doi: 10.1016/j.sysarc.2007.01.007. URL `http://dx.doi.org/10.1016/j.sysarc.2007.01.007`.

[41] Yale Fan. Applications of multi-valued quantum algorithms. *arXiv preprint arXiv:0809.0932*, 2008.

[42] H. Y. Li, C. W. Wu, W. T. Liu, P. X. Chen, and C. Z. Li. Fast quantum search algorithm for databases of arbitrary size and its implementation in a cavity QED system. *Physics Letters A*, 375:4249–4254, November 2011. doi: 10.1016/j.physleta.2011.10.016.

[43] Yushi Wang and Marek Perkowski. Improved complexity of quantum oracles for ternary grover algorithm for graph coloring. In *2011 41st IEEE International Symposium on Multiple-Valued Logic*, pages 294–301. IEEE, 2011.

[44] SS Ivanov, HS Tonchev, and NV Vitanov. Time-efficient implementation of quantum search with qudits. *Physical Review A*, 85(6):062321, 2012.

[45] Alex Bocharov, Martin Roetteler, and Krysta M Svore. Factoring with qutrits: Shor's algorithm on ternary and metaplectic quantum architectures. *Physical Review A*, 96 (1):012306, 2017.

[46] Y. He, M.-X. Luo, E. Zhang, H.-K. Wang, and X.-F. Wang. Decompositions of n-qubit Toffoli Gates with Linear Circuit Complexity. *International Journal of Theoretical Physics*, 56:2350–2361, July 2017. doi: 10.1103/PhysRevA.75.022313.

[47] Benjamin P. Lanyon, Marco Barbieri, Marcelo P. Almeida, Thomas Jennewein, Timothy C. Ralph, Kevin J. Resch, Geoff J. Pryde, Jeremy L. O'Brien, Alexei Gilchrist, and Andrew G. White. Simplifying quantum logic using higher-dimensional hilbert spaces. *Nature Physics*, 5:134 EP –, 12 2008. URL `https://doi.org/10.1038/nphys1150`.

[48] T. C. Ralph, K. J. Resch, and A. Gilchrist. Efficient toffoli gates using qudits. *Phys. Rev. A*, 75:022313, Feb 2007. doi: 10.1103/PhysRevA.75.022313. URL `https://link.aps.org/doi/10.1103/PhysRevA.75.022313`.

[49] Code for asymptotic improvements to quantum circuits via qutrits. `https://github.com/epiqc/qutrits`, 2019.

[50] Francesco Tacchino, Chiara Macchiavello, Dario Gerace, and Daniele Bajoni. An artificial neuron implemented on an actual quantum processor. *npj Quantum Information*, 5 (1):26, 2019.

[51] Francesco Tacchino. Personal Communication.

[52] Thomas G. Draper. Addition on a quantum computer. *arXiv preprint quant-ph/0008033*, 2000.

[53] Craig Gidney. Factoring with n+2 clean qubits and n-1 dirty qubits. *arXiv preprint arXiv:1706.07884*, 2017.

[54] Thomas Häner, Martin Roetteler, and Krysta M. Svore. Factoring using 2n + 2 qubits with toffoli based modular multiplication. *Quantum Info. Comput.*, 17(7-8):673–684, June 2017. ISSN 1533-7146. URL `http://dl.acm.org/citation.cfm?id=3179553.3179560`.

[55] D. G. Cory, M. D. Price, W. Maas, E. Knill, R. Laflamme, W. H. Zurek, T. F. Havel, and S. S. Somaroo. Experimental quantum error correction. *Phys. Rev. Lett.*, 81:2152–2155, Sep 1998. doi: 10.1103/PhysRevLett.81.2152. URL `https://link.aps.org/doi/10.1103/PhysRevLett.81.2152`.

[56] Eric Dennis. Toward fault-tolerant quantum computation without concatenation. *Phys. Rev. A*, 63:052314, Apr 2001. doi: 10.1103/PhysRevA.63.052314. URL `https://link.aps.org/doi/10.1103/PhysRevA.63.052314`.

[57] Matthew Otten and Stephen K Gray. Accounting for errors in quantum algorithms via individual error reduction. *npj Quantum Information*, 5(1):11, 2019.

[58] Daniel Miller, Timo Holz, Hermann Kampermann, and Dagmar Bruß. Propagation of generalized pauli errors in qudit clifford circuits. *Physical Review A*, 98(5):052316, 2018.

[59] N. Khammassi, I. Ashraf, X. Fu, C. G. Almudever, and K. Bertels. Qx: A high-performance quantum computer simulation platform. In *Proceedings of the Conference on Design, Automation & Test in Europe*, DATE '17, pages 464–469, 3001 Leuven, Belgium, Belgium, 2017. European Design and Automation Association. URL `http://dl.acm.org/citation.cfm?id=3130379.3130487`.

[60] Davide Venturelli, Minh Do, Eleanor Rieffel, and Jeremy Frank. Compiling quantum circuits to realistic hardware architectures using temporal planners. *Quantum Science and Technology*, 3(2):025004, 2018.

[61] Kyle EC Booth, Minh Do, J Christopher Beck, Eleanor Rieffel, Davide Venturelli, and Jeremy Frank. Comparing and integrating constraint programming and temporal planning for quantum circuit compilation. In *Twenty-Eighth International Conference on Automated Planning and Scheduling*, 2018.

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

[63] Todd A. Brun. A simple model of quantum trajectories. *American Journal of Physics*, 70(7):719–737, 2002. doi: 10.1119/1.1475328. URL `https://doi.org/10.1119/1.1475328`.

[64] Rüdiger Schack and Todd A Brun. A C++ library using quantum trajectories to solve quantum master equations. *Computer Physics Communications*, 102(1-3):210–228, 1997.

[65] Robert S Smith, Michael J Curtis, and William J Zeng. A practical quantum instruction set architecture. *arXiv preprint arXiv:1608.03355*, 2016.

[66] J Robert Johansson, PD Nation, and Franco Nori. Qutip: An open-source python framework for the dynamics of open quantum systems. *Computer Physics Communications*, 183(8):1760–1772, 2012.

[67] J Robert Johansson, Paul D Nation, and Franco Nori. Qutip 2: A python framework for the dynamics of open quantum systems. *Computer Physics Communications*, 184 (4):1234–1240, 2013.

[68] A. Yu. Chernyavskiy, Vad. V. Voevodin, and Vl. V. Voevodin. Parallel computational structure of noisy quantum circuits simulation. *Lobachevskii Journal of Mathematics*, 39(4):494–502, May 2018. ISSN 1818-9962. doi: 10.1134/S1995080218040042. URL `https://doi.org/10.1134/S1995080218040042`.

[69] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. C. White, J. Mutus, A. G. Fowler, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, C. Neill, P. O'Malley, P. Roushan, A. Vainsencher, J. Wenner, A. N. Korotkov, A. N. Cleland, and John M. Martinis. Superconducting quantum circuits at the surface code threshold for fault tolerance. *Nature*, 508:500 EP –, 04 2014. URL `https://doi.org/10.1038/nature13171`.

[70] Edwin Barnes, Christian Arenz, Alexander Pitchford, and Sophia E. Economou. Fast microwave-driven three-qubit gates for cavity-coupled superconducting qubits. *Phys. Rev. B*, 96:024504, Jul 2017. doi: 10.1103/PhysRevB.96.024504. URL `https://link.aps.org/doi/10.1103/PhysRevB.96.024504`.

[71] Matthew Reagor, Wolfgang Pfaff, Christopher Axline, Reinier W. Heeres, Nissim Ofek, Katrina Sliwa, Eric Holland, Chen Wang, Jacob Blumoff, Kevin Chou, Michael J. Hatridge, Luigi Frunzio, Michel H. Devoret, Liang Jiang, and Robert J. Schoelkopf. Quantum memory with millisecond coherence in circuit qed. *Phys. Rev. B*, 94:014506, Jul 2016. doi: 10.1103/PhysRevB.94.014506. URL `https://link.aps.org/doi/10.1103/PhysRevB.94.014506`.

[72] Nathan Earnest, Srivatsan Chakram, Yao Lu, Nicholas Irons, Ravi K Naik, Nelson Leung, Leo Ocola, David A Czaplewski, Brian Baker, Jay Lawrence, et al. Realization of a λ system with metastable states of a capacitively shunted fluxonium. *Physical review letters*, 120(15):150504, 2018.

[73] Steven M Girvin. Circuit qed: superconducting qubits coupled to microwave photons. *Quantum Machines: Measurement and Control of Engineered Quantum Systems*, page 113, 2011.

[74] Kenneth R Brown, Jungsang Kim, and Christopher Monroe. Co-designing a scalable quantum computer with trapped atomic ions. *npj Quantum Information*, 2:16034, 2016.

[75] Natalie C. Brown and Kenneth R. Brown. Comparing zeeman qubits to hyperfine qubits in the context of the surface code: $^{174}\mathrm{Yb}^+$ and $^{171}\mathrm{Yb}^+$. *Phys. Rev. A*, 97:052301, May 2018. doi: 10.1103/PhysRevA.97.052301. URL `https://link.aps.org/doi/10.1103/PhysRevA.97.052301`.

[76] RK Naik, N Leung, S Chakram, Peter Groszkowski, Y Lu, N Earnest, DC McKay, Jens Koch, and DI Schuster. Random access quantum information processors using multimode circuit quantum electrodynamics. *Nature communications*, 8(1):1–7, 2017.

[77] Mathias Soeken, Stefan Frehse, Robert Wille, and Rolf Drechsler. Revkit: An open source toolkit for the design of reversible circuits. In A. De Vos and R. Wille, editors, *Reversible Computation*, pages 64–76. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-29517-1.

[78] D. M. Miller, D. Maslov, and G. W. Dueck. A transformation based algorithm for reversible logic synthesis. In *Proceedings 2003. Design Automation Conference (IEEE Cat. No.03CH37451)*, pages 318–323, June 2003. doi: 10.1145/775832.775915.

[79] Vahid Karimipour, Azam Mani, and Laleh Memarzadeh. Characterization of qutrit channels in terms of their covariance and symmetry properties. *Phys. Rev. A*, 84:012321, Jul 2011. doi: 10.1103/PhysRevA.84.012321. URL `https://link.aps.org/doi/10.1103/PhysRevA.84.012321`.

[80] Markus Grassl, Linghang Kong, Zhaohui Wei, Zhang-Qi Yin, and Bei Zeng. Quantum error-correcting codes for qudit amplitude damping. *IEEE Transactions on Information Theory*, 64(6):4674–4685, 2018.

[81] Joydip Ghosh, Austin G. Fowler, John M. Martinis, and Michael R. Geller. Understanding the effects of leakage in superconducting quantum-error-detection circuits. *Phys. Rev. A*, 88:062329, Dec 2013. doi: 10.1103/PhysRevA.88.062329. URL `https://link.aps.org/doi/10.1103/PhysRevA.88.062329`.

[82] Joe O'Gorman and Earl T. Campbell. Quantum computation with realistic magic-state factories. *Phys. Rev. A*, 95:032338, Mar 2017. doi: 10.1103/PhysRevA.95.032338. URL `https://link.aps.org/doi/10.1103/PhysRevA.95.032338`.

[83] Martin Suchara, Arvin Faruque, Ching-Yi Lai, Gerardo Paz, Frederic T. Chong, and John Kubiatowicz. Comparing the overhead of topological and concatenated quantum error correction. *arXiv preprint arXiv:1312.2316*, 2013. doi: 10.1109/ICCD.2013.6657074.

[84] 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.

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

[86] Eric R. Anschuetz, Jonathan P. Olson, Alán Aspuru-Guzik, and Yudong Cao. Variational quantum factoring, 2018.

[87] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18 (2):023023, 2016.

[88] Ali JavadiAbhari, Shruti Patil, Daniel Kudrow, Jeff Heckey, Alexey Lvov, Frederic T Chong, and Margaret Martonosi. Scaffcc: a framework for compilation and analysis of quantum computing programs. In *Proceedings of the 11th ACM Conference on Computing Frontiers*, page 1. ACM, 2014.

[89] Daniel Kudrow, Kenneth Bier, Zhaoxia Deng, Diana Franklin, Yu Tomita, Kenneth R. Brown, and Frederic T. Chong. Quantum rotations: A case study in static and dynamic

machine-code generation for quantum computers. In *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ISCA '13, pages 166–176, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2079-5. doi: 10.1145/2485922.2485937. URL `http://doi.acm.org/10.1145/2485922.2485937`.

[90] 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.

[91] Yunseong Nam, Jwo-Sy Chen, Neal C Pisenti, Kenneth Wright, Conor Delaney, Dmitri Maslov, Kenneth R Brown, Stewart Allen, Jason M Amini, Joel Apisdorf, et al. Ground-state energy estimation of the water molecule on a trapped ion quantum computer. *arXiv preprint arXiv:1902.10171*, 2019.

[92] R. Barends, A. Shabani, L. Lamata, J. Kelly, A. Mezzacapo, U. Las Heras, R. Babbush, A. G. Fowler, B. Campbell, Yu Chen, Z. Chen, B. Chiaro, A. Dunsworth, E. Jeffrey, E. Lucero, A. Megrant, J. Y. Mutus, M. Neeley, C. Neill, P. J. J. O'Malley, C. Quintana, P. Roushan, D. Sank, A. Vainsencher, J. Wenner, T. C. White, E. Solano, H. Neven, and John M. Martinis. Digitized adiabatic quantum computing with a superconducting circuit. *Nature*, 534:222 EP –, Jun 2016. URL `https://doi.org/10.1038/nature17658`.

[93] David C McKay, Christopher J Wood, Sarah Sheldon, Jerry M Chow, and Jay M Gambetta. Efficient z gates for quantum computing. *Physical Review A*, 96(2):022330, 2017.

[94] Steffen J. Glaser, Ugo Boscain, Tommaso Calarco, Christiane P. Koch, Walter Köckenberger, Ronnie Kosloff, Ilya Kuprov, Burkhard Luy, Sophie Schirmer, Thomas Schulte-Herbrüggen, Dominique Sugny, and Frank K. Wilhelm. Training schrödinger's cat: quantum optimal control. *The European Physical Journal D*, 69(12):279, Dec 2015. ISSN 1434-6079. doi: 10.1140/epjd/e2015-60464-1. URL `https://doi.org/10.1140/epjd/e2015-60464-1`.

[95] Navin Khaneja, Timo Reiss, Cindie Kehlet, Thomas Schulte-Herbrüggen, and Steffen J. Glaser. Optimal control of coupled spin dynamics: design of nmr pulse sequences by gradient ascent algorithms. *Journal of Magnetic Resonance*, 172(2):296 – 305, 2005. ISSN 1090-7807. doi: https://doi.org/10.1016/j.jmr.2004.11.004. URL `http://www.sciencedirect.com/science/article/pii/S1090780704003696`.

[96] 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. ACM, 2019.

[97] Nelson Leung, Mohamed Abdelhafez, Jens Koch, and David Schuster. Speedup for quantum optimal control from automatic differentiation based on graphics processing units. *Physical Review A*, 95(4):042318, 2017.

[98] Mohamed Abdelhafez, David I. Schuster, and Jens Koch. Gradient-based optimal control of open quantum systems using quantum trajectories and automatic differentiation, 2019.

[99] Yu Chen, C. Neill, P. Roushan, N. Leung, M. Fang, R. Barends, J. Kelly, B. Campbell, Z. Chen, B. Chiaro, A. Dunsworth, E. Jeffrey, A. Megrant, J. Y. Mutus, P. J. J. O'Malley, C. M. Quintana, D. Sank, A. Vainsencher, J. Wenner, T. C. White, Michael R. Geller, A. N. Cleland, and John M. Martinis. Qubit architecture with high coherence and fast tunable coupling. *Phys. Rev. Lett.*, 113:220502, Nov 2014. doi: 10.1103/PhysRevLett.113.220502. URL `https://link.aps.org/doi/10.1103/PhysRevLett.113.220502`.

[100] David C. McKay, Thomas Alexander, Luciano Bello, Michael J. Biercuk, Lev Bishop, Jiayin Chen, Jerry M. Chow, Antonio D. Córcoles, Daniel Egger, Stefan Filipp, Juan Gomez, Michael R. Hush, Ali Javadi-Abhari, Diego Moreda, Paul Nation, Brent Paulovicks, Erick Winston, Christopher J. Wood, James Wootton, and Jay M. Gambetta. Qiskit backend specifications for openqasm and openpulse experiments. *ArXiv*, abs/1809.03452, 2018.

[101] Damian S. Steiger, Thomas Häner, and Matthias Troyer. ProjectQ: an open source software framework for quantum computing. *Quantum*, 2:49, January 2018. ISSN 2521-327X. doi: 10.22331/q-2018-01-31-49. URL `https://doi.org/10.22331/q-2018-01-31-49`.

[102] Krysta M Svore, Alan Geller, Matthias Troyer, John Azariah, Christopher Granade, Bettina Heim, Vadym Kliuchnikov, Mariia Mykhailova, Andres Paz, and Martin Roetteler. Q#: Enabling scalable quantum computing and development with a high-level domain-specific language. *arXiv preprint arXiv:1803.00652*, 2018.

[103] Alexander S Green, Peter LeFanu Lumsdaine, Neil J Ross, Peter Selinger, and Benoît Valiron. Quipper: a scalable quantum programming language. In *ACM SIGPLAN Notices*, volume 48, pages 333–342. ACM, 2013.

[104] Ali Javadi-Abhari, Arvin Faruque, Mohammad Javad Dousti, Lukas Svec, Oana Catu, Amlan Chakrabati, Chen-Fu Chiang, Seth Vanderwilt, John Black, Fred Chong, Margaret Martonosi, Martin Suchara, Ken Brown, Massoud Pedram, and Todd Brun. Scaffold: Quantum programming language, 2012.

[105] P. de Fouquieres, S. G. Schirmer, S. J. Glaser, and I. Kuprov. Second order gradient ascent pulse engineering. *Journal of Magnetic Resonance*, 212:412–417, October 2011. doi: 10.1016/j.jmr.2011.07.023.

[106] Dave Wecker and Krysta M. Svore. Liqui: A software design architecture and domain-specific language for quantum computing, 2014.

[107] 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.

[108] IBM. The qiskit roadmap 2019. https://github.com/Qiskit/qiskit/blob/master/docs/development_strategy.rst, 2019.

[109] Sam McArdle, Suguru Endo, Alan Aspuru-Guzik, Simon Benjamin, and Xiao Yuan. Quantum computational chemistry. *arXiv preprint arXiv:1808.10402*, 2018.

[110] Henry Eyring. The activated complex in chemical reactions. *The Journal of Chemical Physics*, 3(2):107–115, 1935.

[111] Hauke Paulsen and Alfred X Trautwein. Density functional theory calculations for spin crossover complexes. In *Spin Crossover in Transition Metal Compounds III*, pages 197–219. Springer, 2004.

[112] Seth Lloyd. Universal quantum simulators. *Science*, pages 1073–1078, 1996.

[113] Daochen Wang, Oscar Higgott, and Stephen Brierley. A generalised variational quantum eigensolver. *arXiv preprint arXiv:1802.00171*, 2018.

[114] Jonathan Romero, Ryan Babbush, Jarrod R McClean, Cornelius Hempel, Peter J Love, and Alán Aspuru-Guzik. Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz. *Quantum Science and Technology*, 4(1):014008, 2018.

[115] Rodney J Bartlett and Monika Musiał. Coupled-cluster theory in quantum chemistry. *Reviews of Modern Physics*, 79(1):291, 2007.

[116] Panagiotis Kl Barkoutsos, Jerome F Gonthier, Igor Sokolov, Nikolaj Moll, Gian Salis, Andreas Fuhrer, Marc Ganzhorn, Daniel J Egger, Matthias Troyer, Antonio Mezzacapo, et al. Quantum algorithms for electronic structure calculations: Particle-hole hamiltonian and optimized wave-function expansions. *Physical Review A*, 98(2):022322, 2018.

[117] Qiming Sun, Timothy C Berkelbach, Nick S Blunt, George H Booth, Sheng Guo, Zhendong Li, Junzi Liu, James D McClain, Elvira R Sayfutyarova, Sandeep Sharma, et al. Pyscf: the python-based simulations of chemistry framework. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 8(1):e1340, 2018.

[118] Ryan Babbush, Jarrod McClean, Dave Wecker, Alán Aspuru-Guzik, and Nathan Wiebe. Chemical basis of trotter-suzuki errors in quantum chemistry simulation. *Phys. Rev. A*, 91:022311, Feb 2015. doi: 10.1103/PhysRevA.91.022311. URL `https://link.aps.org/doi/10.1103/PhysRevA.91.022311`.

[119] Pranav Gokhale, Yongshan Ding, Thomas Propson, and Christopher Winkler. Code and results: Partial compilation of variational algorithms. `https://github.com/EPiQC/PartialCompilation`, Aug 2019.

[120] Seth Lloyd. Quantum approximate optimization is computationally universal, 2018.

[121] J. S. Otterbach, R. Manenti, N. Alidoust, A. Bestwick, M. Block, B. Bloom, S. Caldwell, N. Didier, E. Schuyler Fried, S. Hong, P. Karalekas, C. B. Osborn, A. Papageorge, E. C. Peterson, G. Prawiroatmodjo, N. Rubin, Colm A. Ryan, D. Scarabelli, M. Scheer, E. A. Sete, P. Sivarajah, Robert S. Smith, A. Staley, N. Tezak, W. J. Zeng, A. Hudson, Blake R. Johnson, M. Reagor, M. P. da Silva, and C. Rigetti. Unsupervised machine learning on a hybrid quantum computer, 2017.

[122] Edward Farhi and Aram W Harrow. Quantum supremacy through the quantum approximate optimization algorithm, 2016.

[123] Florian Dolde, Ville Bergholm, Ya Wang, Ingmar Jakobi, Boris Naydenov, Sébastien Pezzagna, Jan Meijer, Fedor Jelezko, Philipp Neumann, Thomas Schulte-Herbrüggen, Jacob Biamonte, and Jörg Wrachtrup. High-fidelity spin entanglement using optimal control. *Nature Communications*, 5:3371 EP –, Feb 2014. URL `https://doi.org/10.1038/ncomms4371`. Article.

[124] Yi Chou, Shang-Yu Huang, and Hsi-Sheng Goan. Optimal control of fast and high-fidelity quantum gates with electron and nuclear spins of a nitrogen-vacancy center in diamond. *Phys. Rev. A*, 91:052315, May 2015. doi: 10.1103/PhysRevA.91.052315. URL `https://link.aps.org/doi/10.1103/PhysRevA.91.052315`.

[125] Seth Lloyd and Reevu Maity. Efficient implementation of unitary transformations, 2019.

[126] X. Fu, L. Riesebos, M. A. Rol, J. van Straten, J. van Someren, N. Khammassi, I. Ashraf, R. F. L. Vermeulen, V. Newsum, K. K. L. Loh, J. C. de Sterke, W. J. Vlothuizen, R. N. Schouten, C. G. Almudever, L. DiCarlo, and K. Bertels. eqasm: An executable quantum instruction set architecture. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 224–237, Feb 2019. doi: 10.1109/HPCA.2019.00040.

[127] Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855. ACM, 2013.

[128] Gonzalo I Diaz, Achille Fokoue-Nkoutche, Giacomo Nannicini, and Horst Samulowitz. An effective algorithm for hyperparameter optimization of neural networks. *IBM Journal of Research and Development*, 61(4/5):9–1, 2017.

[129] Giacomo Nannicini. Performance of hybrid quantum-classical variational heuristics for combinatorial optimization. *Physical Review E*, 99(1):013304, 2019.

[130] A Yu Kitaev. Quantum measurements and the abelian stabilizer problem. *arXiv preprint quant-ph/9511026*, 1995.

[131] 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.

[132] Tzvetan S Metodi, Arvin I Faruque, and Frederic T Chong. Quantum computing for computer architects. *Synthesis Lectures on Computer Architecture*, 6(1):1–203, 2011.

[133] Andy Matuschak and Michael A Nielsen. Quantum computing for the very curiuos. 2019.

[134] Microsoft. Pauli measurements (q# docs), Dec 2017.

[135] Ramamurti Shankar. *Principles of quantum mechanics*. Springer Science & Business Media, 2012.

[136] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[137] Douglas L. Strout and Gustavo E. Scuseria. A quantitative study of the scaling properties of the hartree–fock method. *The Journal of Chemical Physics*, 102(21): 8448–8452, 1995. doi: 10.1063/1.468836.

[138] G. Kresse and J. Furthmüller. Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set. *Computational Materials Science*, 6 (1):15 – 50, 1996. ISSN 0927-0256. doi: https://doi.org/10.1016/0927-0256(96)00008-0.

[139] Gustavo E. Scuseria and Timothy J. Lee. Comparison of coupled-cluster methods which include the effects of connected triple excitations. *The Journal of Chemical Physics*, 93 (8):5851–5855, 1990. doi: 10.1063/1.459684.

[140] C. David Sherrill. Computational scaling of the configuration interaction method with system size. 1996. URL `http://vergil.chemistry.gatech.edu/notes/ciscale/ciscale.html`.

[141] Kenji Sugisaki, Shigeaki Nakazawa, Kazuo Toyota, Kazunobu Sato, Daisuke Shiomi, and Takeji Takui. Quantum chemistry on quantum computers: A method for preparation of multiconfigurational wave functions on quantum computers without performing post-hartree–fock calculations. *ACS Central Science*, 5(1):167–175, 2018.

[142] Robert M Parrish, Lori A Burns, Daniel GA Smith, Andrew C Simmonett, A Eugene DePrince III, Edward G Hohenstein, Ugur Bozkaya, Alexander Yu Sokolov, Roberto Di Remigio, Ryan M Richard, et al. Psi4 1.1: An open-source electronic structure program emphasizing automation, advanced libraries, and interoperability. *Journal of chemical theory and computation*, 13(7):3185–3197, 2017.

[143] Jarrod R McClean, Kevin J Sung, Ian D Kivlichan, Yudong Cao, Chengyu Dai, E Schuyler Fried, Craig Gidney, Brendan Gimby, Pranav Gokhale, Thomas Häner, et al.

Openfermion: the electronic structure package for quantum computers. *arXiv preprint arXiv:1710.07629*, 2017.

[144] Dave Wecker, Bela Bauer, Bryan K Clark, Matthew B Hastings, and Matthias Troyer. Gate-count estimates for performing quantum chemistry on small quantum computers. *Physical Review A*, 90(2):022305, 2014.

[145] Matthew B Hastings, Dave Wecker, Bela Bauer, and Matthias Troyer. Improving quantum algorithms for quantum chemistry. *arXiv preprint arXiv:1403.1539*, 2014.

[146] Pascual Jordan and Eugene P Wigner. About the pauli exclusion principle. *Z. Phys.*, 47:631–651, 1928.

[147] Jacob T Seeley, Martin J Richard, and Peter J Love. The bravyi-kitaev transformation for quantum computation of electronic structure. *The Journal of chemical physics*, 137 (22):224109, 2012.

[148] Sergey B Bravyi and Alexei Yu Kitaev. Fermionic quantum computation. *Annals of Physics*, 298(1):210–226, 2002.

[149] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):4812, 2018.

[150] Alexander J McCaskey, Zachary P Parks, Jacek Jakowski, Shirley V Moore, T Morris, Travis S Humble, and Raphael C Pooser. Quantum chemistry as a benchmark for near-term quantum computers. *arXiv preprint arXiv:1905.01534*, 2019.

[151] Cornelius Hempel, Christine Maier, Jonathan Romero, Jarrod McClean, Thomas Monz, Heng Shen, Petar Jurcevic, Ben P Lanyon, Peter Love, Ryan Babbush, et al. Quantum chemistry calculations on a trapped-ion quantum simulator. *Physical Review X*, 8(3): 031022, 2018.

[152] Joonho Lee, William J Huggins, Martin Head-Gordon, and K Birgitta Whaley. Generalized unitary coupled cluster wave functions for quantum computation. *Journal of chemical theory and computation*, 15(1):311–324, 2018.

[153] Julian Schwinger. Unitary operator bases. *Proceedings of the national academy of sciences of the United States Of America*, 46(4):570, 1960.

[154] Andreas Klappenecker and Martin Rötteler. Constructions of mutually unbiased bases. In *International Conference on Finite Fields and Applications*, pages 137–144. Springer, 2003.

[155] Pranav Gokhale, Olivia Angiuli, Yongshan Ding, Kaiwen Gui, Teague Tomesh, Margaret Martonosi, and Frederic T Chong. Minimizing state preparations for vqe. Quantum Resource Estimation Workshop, Dec 2019. URL `https://www.quantumresource.org/pdfs/gokhale.pdf`.

[156] Andrew Jena, Scott Genin, and Michele Mosca. Pauli partitioning with respect to gate sets. *arXiv preprint arXiv:1907.07859*, 2019.

[157] Vladyslav Verteletskyi, Tzu-Ching Yen, and Artur F Izmaylov. Measurement optimization in the variational quantum eigensolver using a minimum clique cover. *arXiv preprint arXiv:1907.03358*, 2019.

[158] Tzu-Ching Yen, Vladyslav Verteletskyi, and Artur F Izmaylov. Measuring all compatible operators in one series of single-qubit measurements using unitary transformations. *arXiv preprint arXiv:1907.09386*, 2019.

[159] Artur F Izmaylov, Tzu-Ching Yen, Robert A Lang, and Vladyslav Verteletskyi. Unitary partitioning approach to the measurement problem in the variational quantum eigensolver method. *arXiv preprint arXiv:1907.09040*, 2019.

[160] Christian Kokail, Christine Maier, Rick van Bijnen, Tiff Brydges, Manoj K Joshi, Petar Jurcevic, Christine A Muschik, Pietro Silvi, Rainer Blatt, Christian F Roos, et al. Self-verifying variational quantum simulation of the lattice schwinger model. *arXiv preprint arXiv:1810.03421*, 2018.

[161] Sergey Bravyi, Jay M Gambetta, Antonio Mezzacapo, and Kristan Temme. Tapering off qubits to simulate fermionic hamiltonians. *arXiv preprint arXiv:1701.08213*, 2017.

[162] Artur F Izmaylov, Tzu-Ching Yen, and Ilya G Ryabinkin. Revising the measurement process in the variational quantum eigensolver: is it possible to reduce the number of separately measured operators? *Chemical Science*, 2019.

[163] B Moseley, M Osborne, and S Benjamin. Bayesian optimisation for variational quantum eigensolvers. *quantum*, 3:4, 2018.

[164] Pierre-Luc Dallaire-Demers, Jonathan Romero, Libor Veis, Sukin Sim, and Alán Aspuru-Guzik. Low-depth circuit ansatz for preparing correlated fermionic states on a quantum computer. *arXiv preprint arXiv:1801.01053*, 2018.

[165] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[166] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 681–690. ACM, 2006.

[167] Michel Planat and Metod Saniga. On the pauli graphs of n-qudits. *arXiv preprint quant-ph/0701211*, 2007.

[168] Ravi Boppana and Magnús M Halldórsson. Approximating maximum independent sets by excluding subgraphs. *BIT Numerical Mathematics*, 32(2):180–196, 1992.

[169] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[170] Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973.

[171] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL `http://www.scipy.org/`. [Online; accessed ¡today¿].

[172] James D Whitfield, Jacob Biamonte, and Alán Aspuru-Guzik. Simulation of electronic structure hamiltonians using quantum computers. *Molecular Physics*, 109(5):735–750, 2011.

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

[174] Navin Khaneja and Steffen Glaser. Cartan decomposition of su (2ˆ n), constructive controllability of spin systems and universal quantum computing. *arXiv preprint quant-ph/0010100*, 2000.

[175] Neil J Ross. Optimal ancilla-free pauli+ v approximation of z-rotations. *arXiv preprint arXiv:1409.4355*, 2014.

[176] Chi-Kwong Li, Rebecca Roberts, and Xiaoyan Yin. Decomposition of unitary matrices and quantum gates. *International Journal of Quantum Information*, 11(01):1350015, 2013.

[177] Anmer Daskin and Sabre Kais. Decomposition of unitary matrices for finding quantum circuits: application to molecular hamiltonians. *The Journal of chemical physics*, 134 (14):144112, 2011.

[178] Yumi Nakajima, Yasuhito Kawano, and Hiroshi Sekigawa. A new algorithm for producing quantum circuits using kak decompositions. *arXiv preprint quant-ph/0509196*, 2005.

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

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

[181] Ulrich Seyfarth and Kedar S Ranade. Construction of mutually unbiased bases with cyclic symmetry for qubit systems. *Physical Review A*, 84(4):042327, 2011.

[182] Ulrich Seyfarth. Cyclic mutually unbiased bases and quantum public-key encryption. *arXiv preprint arXiv:1907.02726*, 2019.

[183] Yi-Cong Zheng, Ching-Yi Lai, Todd A Brun, and Leong-Chuan Kwek. Depth reduction for quantum clifford circuits through pauli measurements. *arXiv preprint arXiv:1805.12082*, 2018.

[184] Richard William Farebrother. *Linear least squares computations*. Marcel Dekker, Inc., 1988.

[185] Eugene F Dumitrescu, Alex J McCaskey, Gaute Hagen, Gustav R Jansen, Titus D Morris, T Papenbrock, Raphael C Pooser, David Jarvis Dean, and Pavel Lougovski. Cloud quantum computing of an atomic nucleus. *Physical review letters*, 120(21):210501, 2018.

[186] Karol Życzkowski, Paweł Horodecki, Anna Sanpera, and Maciej Lewenstein. Volume of the set of separable states. *Physical Review A*, 58(2):883, 1998.

[187] Nicholas J Russell, Levon Chakhmakhchyan, Jeremy L O'Brien, and Anthony Laing. Direct dialling of haar random unitary matrices. *New journal of physics*, 19(3):033007, 2017.

[188] Alán Aspuru-Guzik, Anthony D Dutoi, Peter J Love, and Martin Head-Gordon. Simulated quantum computation of molecular energies. *Science*, 309(5741):1704–1707, 2005.

[189] Ophelia Crawford, Barnaby van Straaten, Daochen Wang, Thomas Parks, Earl Campbell, and Stephen Brierley. Efficient quantum measurement of pauli operators. *arXiv preprint arXiv:1908.06942*, 2019.

[190] William J Huggins, Jarrod McClean, Nicholas Rubin, Zhang Jiang, Nathan Wiebe, K Birgitta Whaley, and Ryan Babbush. Efficient and noise resilient measurements for quantum chemistry on near-term quantum computers. *arXiv preprint arXiv:1907.13117*, 2019.

[191] Andrew Zhao, Andrew Tranter, William M Kirby, Shu Fay Ung, Akimasa Miyake, and Peter Love. Measurement reduction in variational quantum algorithms. *arXiv preprint arXiv:1908.08067*, 2019.

[192] Robert F Bailey and Brett Stevens. Hamiltonian decompositions of complete k-uniform hypergraphs. *Discrete Mathematics*, 310(22):3088–3095, 2010.

[193] Ulrich Tamm. Applications of baranyai's theorem in information theory.

[194] Narsingh Deo and Paulius Micikevicius. On one-factorization of complete 3-uniform hypergraphs. *Congressus Numerantium*, pages 153–162, 2002.

[195] AE Brouwer and A Schrijver. Uniform hypergraphs. *Packing and Covering in Combinatorics (A. Schrijver, ed.)*, pages 39–93, 1979.

[196] Eric W. Weisstein. Baranyai's theorem. From MathWorld—A Wolfram Web Resource. URL `http://mathworld.wolfram.com/BaranyaisTheorem.html`.

[197] Padraic Bartlett. Lecture 2: The max-flow min-cut theorem, April 2015.

[198] David Eisenstat. Find unique compositions of k-distinct-element subsets for a set of n elements. Stack Overflow. URL `https://stackoverflow.com/a/25902878`. URL:https://stackoverflow.com/a/25902878 (version: 2019-08-06).

[199] LR Ford and DR Fulkerson. Maximal flow through a network. 1954.

[200] Edith Cohen. Approximate max-flow on small depth networks. *SIAM Journal on Computing*, 24(3):579–597, 1995.

[201] Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 253–262. IEEE, 2013.

[202] Yin Tat Lee, Satish Rao, and Nikhil Srivastava. A new approach to computing maximum flows using electrical flows. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 755–764. ACM, 2013.

[203] Donggu Kang and James Payor. Flow rounding. *arXiv preprint arXiv:1507.08139*, 2015.

[204] Eric R Anschuetz and Cristian Zanoci. Near-term quantum-classical associative adversarial networks. *arXiv preprint arXiv:1905.13205*, 2019.

[205] Chao Song, Kai Xu, Hekang Li, Yu-Ran Zhang, Xu Zhang, Wuxin Liu, Qiujiang Guo, Zhen Wang, Wenhui Ren, Jie Hao, et al. Generation of multicomponent atomic schrodinger cat states of up to 20 qubits. *Science*, 365(6453):574–577, 2019.

[206] Ahmed Omran, H. Levine, A. Keesling, G. Semeghini, T. T. Wang, S. Ebadi, H. Bernien, A. Zibrov, H. Pichler, S. Choi, J. Cui, M. Rossignolo, P. Rembold, S. Montangero, T. Calarco, M. Endres, M. Greiner, V. Vuletić, and M. D. Lukin. Generation and manipulation of schrödinger cat states in rydberg atom arrays. *Science*, 365:570 – 574, 2019.

[207] Thomas Alexander, Naoki Kanazawa, Daniel J Egger, Lauren Capelluto, Christopher J Wood, Ali Javadi-Abhari, and David McKay. Qiskit pulse: Programming quantum computers through the cloud with pulses. *arXiv preprint arXiv:2004.06755*, 2020.

[208] JWO Garmon, RC Pooser, and EF Dumitrescu. Benchmarking noise extrapolation with openpulse. *arXiv preprint arXiv:1909.05219*, 2019.

[209] Ibm q experience. `https://quantum-computing.ibm.com/`, 2019.

[210] Optimizations via openpulse repo. github.com/singular-value/optimizations_via_openpulse, 2020.

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

[212] Prakash Murali, Jonathan M Baker, Ali Javadi-Abhari, Frederic T Chong, and Margaret Martonosi. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1015–1029. ACM, 2019.

[213] Swamit S Tannu and Moinuddin K Qureshi. Not all qubits are created equal: a case for variability-aware policies for nisq-era quantum computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 987–999. ACM, 2019.

[214] J Majer, JM Chow, JM Gambetta, Jens Koch, BR Johnson, JA Schreier, L Frunzio, DI Schuster, Andrew Addison Houck, Andreas Wallraff, et al. Coupling superconducting qubits via a cavity bus. *Nature*, 449(7161):443, 2007.

[215] YM Galperin, DV Shantsev, J Bergli, and BL Altshuler. Rabi oscillations of a qubit coupled to a two-level system. *EPL (Europhysics Letters)*, 71(1):21, 2005.

[216] S Ashhab, JR Johansson, and Franco Nori. Rabi oscillations in a qubit coupled to a quantum two-level system. *New Journal of Physics*, 8(6):103, 2006.

[217] Felix Motzoi, Jay M Gambetta, Patrick Rebentrost, and Frank K Wilhelm. Simple pulses for elimination of leakage in weakly nonlinear qubits. *Physical review letters*, 103 (11):110501, 2009.

[218] Jay M Gambetta, F Motzoi, ST Merkel, and Frank K Wilhelm. Analytic control methods for high-fidelity unitary operations in a weakly nonlinear oscillator. *Physical Review A*, 83(1):012308, 2011.

[219] Felix Motzoi and Frank K Wilhelm. Improving frequency selection of driven pulses using derivative-based transition suppression. *Physical Review A*, 88(6):062318, 2013.

[220] Barbara Jones and Maria Vyushkova. Quantum computers flip the script on spin chemistry. `https://www.ibm.com/blogs/research/2020/02/quantum-spin-chemistry/`, 2020.

[221] Filip B Maciejewski, Zoltán Zimborás, and Michał Oszmaniec. Mitigation of readout noise in near-term quantum devices by classical post-processing based on detector tomography. *arXiv preprint arXiv:1907.08518*, 2019.

[222] Yanzhu Chen, Maziar Farahzad, Shinjae Yoo, and Tzu-Chieh Wei. Detector tomography on ibm 5-qubit quantum computers and mitigation of imperfect measurement. *arXiv preprint arXiv:1904.11935*, 2019.

[223] Abraham Asfaw, Thomas Alexander, Paul Nation, and Jay Gambetta. Get to the heart of real quantum hardware. `https://www.ibm.com/blogs/research/2019/12/qiskit-openpulse/`, 2019.

[224] Héctor Abraham, Ismail Yunus Akhalwaya, Gadi Aleksandrowicz, Thomas Alexander, Gadi Alexandrowics, Eli Arbel, Abraham Asfaw, Carlos Azaustre, AzizNgoueya, Panagiotis Barkoutsos, George Barron, Luciano Bello, Yael Ben-Haim, Daniel Bevenius, Lev S. Bishop, Sorin Bolos, Samuel Bosch, Sergey Bravyi, David Bucher, Fran Cabrera, Padraic Calpin, Lauren Capelluto, Jorge Carballo, Ginés Carrascal, Adrian Chen, Chun-Fu Chen, Richard Chen, Jerry M. Chow, Christian Claus, Christian Clauss, Abigail J. Cross, Andrew W. Cross, Simon Cross, Juan Cruz-Benito, Chris Culver, Antonio D. Córcoles-Gonzales, Sean Dague, Tareq El Dandachi, Matthieu Dartiailh, DavideFrr, Abdón Rodríguez Davila, Delton Ding, Jun Doi, Eric Drechsler, Drew, Eugene Dumitrescu, Karel Dumon, Ivan Duran, Kareem EL-Safty, Eric Eastman, Pieter Eendebak, Daniel Egger, Mark Everitt, Paco Martín Fernández, Axel Hernández Ferrera, Albert Frisch, Andreas Fuhrer, MELVIN GEORGE, Julien Gacon, Gadi, Borja Godoy Gago, Jay M. Gambetta, Adhisha Gammanpila, Luis Garcia, Shelly Garion, Juan Gomez-Mosquera, Salvador de la Puente González, Jesse Gorzinski, Ian Gould, Donny Greenberg, Dmitry Grinko, Wen Guan, John A. Gunnels, Mikael Haglund, Isabel Haide, Ikko Hamamura, Vojtech Havlicek, Joe Hellmers, Łukasz Herok, Stefan Hillmich, Hiroshi Horii, Connor Howington, Shaohan Hu, Wei Hu, Haruki Imai, Takashi Imamichi, Kazuaki Ishizaki, Raban Iten, Toshinari Itoko, Ali Javadi, Ali Javadi-Abhari, Jessica, Kiran Johns, Tal Kachmann, Naoki Kanazawa, Kang-Bae, Anton Karazeev, Paul Kassebaum, Spencer King, Knabberjoe, Arseny Kovyrshin, Rajiv Krishnakumar, Vivek Krishnan, Kevin Krsulich, Gawel Kus, Ryan LaRose, Raphaël Lambert, Joe Latone, Scott Lawrence, Dennis Liu, Peng Liu, Yunho Maeng, Aleksei Malyshev, Jakub Marecek, Manoel Marques, Dolph Mathews, Atsushi Matsuo, Douglas T. McClure, Cameron McGarry, David McKay, Dan McPherson, Srujan Meesala, Martin Mevissen, Antonio Mezzacapo, Rohit Midha, Zlatko Minev, Abby Mitchell, Nikolaj Moll, Michael Duane Mooring, Renier Morales, Niall Moran, Prakash Murali, Jan Müggenburg, David Nadlinger, Ken Nakanishi, Giacomo Nannicini, Paul Nation, Yehuda Naveh, Patrick Neuweiler, Pradeep Niroula, Hassi Norlen, Lee James O'Riordan, Oluwatobi Ogunbayo, Pauline Ollitrault, Steven Oud, Dan Padilha, Hanhee Paik, Simone Perriello, Anna Phan, Francesco Piro, Marco Pistoia, Alejandro Pozas-iKerstjens, Viktor Prutyanov, Daniel Puzzuoli, Jesús Pérez, Quintiii, Rudy Raymond, Rafael Martín-Cuevas Redondo, Max Reuter, Julia Rice, Diego M. Rodríguez, RohithKarur, Max Rossmannek, Mingi Ryu, Tharrmashastha SAPV, SamFerracin, Martin Sandberg, Hayk Sargsyan, Ninad Sathaye, Bruno Schmitt, Chris Schnabel, Zachary Schoenfeld, Travis L. Scholten, Eddie Schoute, Joachim Schwarm, Ismael Faro Sertage, Kanav Setia, Nathan Shammah, Yunong Shi, Adenilton Silva, Andrea Simonetto, Nick Singstock, Yukio Siraichi, Iskan-

dar Sitdikov, Seyon Sivarajah, Magnus Berg Sletfjerding, John A. Smolin, Mathias Soeken, Igor Olegovich Sokolov, SooluThomas, Dominik Steenken, Matt Stypulkoski, Jack Suen, Kevin J. Sung, Hitomi Takahashi, Ivano Tavernelli, Charles Taylor, Pete Taylour, Soolu Thomas, Mathieu Tillet, Maddy Tod, Enrique de la Torre, Kenso Trabing, Matthew Treinish, TrishaPe, Wes Turner, Yotam Vaknin, Carmen Recio Valcarce, Francois Varchon, Almudena Carrera Vazquez, Desiree Vogt-Lee, Christophe Vuillot, James Weaver, Rafal Wieczorek, Jonathan A. Wildstrom, Robert Wille, Erick Winston, Jack J. Woehr, Stefan Woerner, Ryan Woo, Christopher J. Wood, Ryan Wood, Steve Wood, James Wootton, Daniyar Yeralin, Richard Young, Jessie Yu, Christopher Zachow, Laura Zdanski, Christa Zoufal, Zoufalc, a matsuo, azulehner, bcamorrison, brandhsn, chlorophyll zz, dan1pal, dime10, drholmie, elfrocampeador, enavarro51, faisaldebouni, fanizzamarco, gadial, gruu, kanejess, klinvill, kurarrr, lerongil, ma5x, merav aharoni, michelle4654, ordmoj, sethmerkel, strickroman, sumitpuri, tigerjack, toural, vvilpas, welien, willhbang, yang.luh, yelojakit, and yotamvakninibm. Qiskit: An open-source framework for quantum computing, 2019.

[225] Ali J Abhari, Arvin Faruque, Mohammad J Dousti, Lukas Svec, Oana Catu, Amlan Chakrabati, Chen-Fu Chiang, Seth Vanderwilt, John Black, and Fred Chong. Scaffold: Quantum programming language. Technical report, Princeton University Department of Computer Science, 2012.

[226] Anders Sørensen and Klaus Mølmer. Entanglement and quantum computation with ions in thermal motion. *Physical Review A*, 62(2):022311, 2000.

[227] Dmitri Maslov and Yunseong Nam. Use of global interactions in efficient quantum circuit constructions. *New Journal of Physics*, 20(3):033018, 2018.

[228] N Khammassi, GG Guerreschi, I Ashraf, JW Hogaboam, CG Almudever, and K Bertels. cqasm v1. 0: Towards a common quantum assembly language. *arXiv preprint arXiv:1805.09607*, 2018.

[229] Nathan Killoran, Josh Izaac, Nicolás Quesada, Ville Bergholm, Matthew Amy, and Christian Weedbrook. Strawberry fields: A software platform for photonic quantum computing. *Quantum*, 3:129, 2019.

[230] Philip Krantz, Morten Kjaergaard, Fei Yan, Terry P Orlando, Simon Gustavsson, and William D Oliver. A quantum engineer's guide to superconducting qubits. *Applied Physics Reviews*, 6(2):021318, 2019.

[231] Navin Khaneja and Steffen J Glaser. Cartan decomposition of su (2n) and control of spin systems. *Chemical Physics*, 267(1-3):11–23, 2001.

[232] Andrew W Cross, Lev S Bishop, Sarah Sheldon, Paul D Nation, and Jay M Gambetta. Validating quantum computers using randomized model circuits. *Physical Review A*, 100(3):032328, 2019.

[233] GS Paraoanu. Microwave-induced coupling of superconducting qubits. *Physical Review B*, 74(14):140504, 2006.

[234] Chad Rigetti and Michel Devoret. Fully microwave-tunable universal gates in superconducting qubits with linear couplings and fixed transition frequencies. *Physical Review B*, 81(13):134507, 2010.

[235] Jerry M Chow, AD Córcoles, Jay M Gambetta, Chad Rigetti, BR Johnson, John A Smolin, JR Rozen, George A Keefe, Mary B Rothwell, Mark B Ketchen, et al. Simple all-microwave entangling gate for fixed-frequency superconducting qubits. *Physical review letters*, 107(8):080502, 2011.

[236] S Poletto, Jay M Gambetta, Seth T Merkel, John A Smolin, Jerry M Chow, AD Córcoles, George A Keefe, Mary B Rothwell, JR Rozen, DW Abraham, et al. Entanglement of two superconducting qubits in a waveguide cavity via monochromatic two-photon excitation. *Physical review letters*, 109(24):240505, 2012.

[237] Jerry M Chow, Jay M Gambetta, Andrew W Cross, Seth T Merkel, Chad Rigetti, and M Steffen. Microwave-activated conditional-phase gate for superconducting qubits. *New Journal of Physics*, 15(11):115012, 2013.

[238] Jerry Moy Chow. *Quantum information processing with superconducting qubits*. Yale University, 2010.

[239] Frederick W Strauch, Philip R Johnson, Alex J Dragt, CJ Lobb, JR Anderson, and FC Wellstood. Quantum logic gates for coupled superconducting phase qubits. *Physical review letters*, 91(16):167005, 2003.

[240] Leonardo DiCarlo, Jerry M Chow, Jay M Gambetta, Lev S Bishop, Blake R Johnson, DI Schuster, J Majer, Alexandre Blais, L Frunzio, SM Girvin, et al. Demonstration of two-qubit algorithms with a superconducting quantum processor. *Nature*, 460(7252): 240, 2009.

[241] Daniel Loss and David P DiVincenzo. Quantum computation with quantum dots. *Physical Review A*, 57(1):120, 1998.

[242] Dima Mozyrsky, Vladimir Privman, and M Lawrence Glasser. Indirect interaction of solid-state qubits via two-dimensional electron gas. *Physical review letters*, 86(22):5112, 2001.

[243] P Echternach, Colin P Williams, SC Dultz, P Delsing, SL Braunstein, and JP Dowling. Universal quantum gates for single cooper pair box based quantum computing. *arXiv preprint quant-ph/0112025*, 2001.

[244] AV Lebedev, GB Lesovik, VM Vinokur, and G Blatter. Extended quantum maxwell demon acting over macroscopic distances. *Physical Review B*, 98(21):214502, 2018.

[245] Michael JD Powell. A view of algorithms for optimization without derivatives. *Mathematics Today-Bulletin of the Institute of Mathematics and its Applications*, 43(5): 170–174, 2007.

[246] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python. *arXiv e-prints*, art. arXiv:1907.10121, Jul 2019.

[247] M Müller, Klemens Hammerer, YL Zhou, Christian F Roos, and P Zoller. Simulating open quantum systems: from many-body interactions to stabilizer pumping. *New Journal of Physics*, 13(8):085007, 2011.

[248] Ali Javadi-Abhari, Paul Nation, and Jay Gambetta. Qiskit – write once, target multiple architectures. https://www.ibm.com/blogs/research/2019/11/qiskit-for-multiple-architectures/, 2019.

[249] Thien Nguyen and Alexander McCaskey. Enabling pulse-level programming, compilation, and execution in xacc. *arXiv preprint arXiv:2003.11971*, 2020.

[250] Alexander Joseph McCaskey, Dmitry Lyakh, Eugene Dumitrescu, Sarah Powers, and Travis S Humble. Xacc: a system-level software infrastructure for heterogeneous quantum-classical computing. *Quantum Science and Technology*, 2020.

[251] J Werschnik and EKU Gross. Quantum optimal control theory. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 40(18):R175, 2007.

[252] Jerry M. Chow, Leonardo DiCarlo, Jay M. Gambetta, Felix Motzoi, Luigi Frunzio, S M Girvin, and Robert J. Schoelkopf. Implementing optimal control pulse shaping for improved single-qubit gates, 2010.

[253] Zhaoqi Leng, Pranav Mundada, Saeed Ghadimi, and Andrew Houck. Robust and efficient algorithms for high-dimensional black-box quantum optimization. *arXiv preprint arXiv:1910.03591*, 2019.

[254] 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.

[255] Sarah Sheldon, Easwar Magesan, Jerry M Chow, and Jay M Gambetta. Procedure for systematically tuning up cross-talk in the cross-resonance gate. *Physical Review A*, 93 (6):060302, 2016.

[256] Antonio D Córcoles, Jay M Gambetta, Jerry M Chow, John A Smolin, Matthew Ware, Joel Strand, Britton LT Plourde, and Matthias Steffen. Process verification of two-qubit quantum gates by randomized benchmarking. *Physical Review A*, 87(3):030301, 2013.

[257] Easwar Magesan and Jay M Gambetta. Effective hamiltonian models of the cross-resonance gate. *arXiv preprint arXiv:1804.04073*, 2018.

[258] Michael J Peterer, Samuel J Bader, Xiaoyue Jin, Fei Yan, Archana Kamal, Theodore J Gudmundsen, Peter J Leek, Terry P Orlando, William D Oliver, and Simon Gustavsson. Coherence and decay of higher energy levels of a superconducting transmon qubit. *Physical review letters*, 114(1):010501, 2015.

[259] Hugo Ribeiro and Aashish A Clerk. Accelerated adiabatic quantum gates: optimizing speed versus robustness. *arXiv preprint arXiv:1906.06737*, 2019.

[260] HY Li, CW Wu, WT Liu, PX Chen, and CZ Li. Fast quantum search algorithm for databases of arbitrary size and its implementation in a cavity qed system. *Physics Letters A*, 375(48):4249–4254, 2011.

[261] Sam McArdle, Xiao Yuan, and Simon Benjamin. Error-mitigated digital quantum simulation. *Physical review letters*, 122(18):180501, 2019.

[262] Serge Rosenblum, P Reinhold, Mazyar Mirrahimi, Liang Jiang, L Frunzio, and RJ Schoelkopf. Fault-tolerant detection of a quantum error. *Science*, 361(6399): 266–270, 2018.

[263] Zafer Gedik, Isabela Almeida Silva, Barış Çakmak, Göktug Karpat, Edson Luiz Géa Vidoto, Diogo de Oliveira Soares-Pinto, ER Deazevedo, and Felipe Fernandes Fanchini. Computational speed-up with a single qudit. *Scientific reports*, 5:14671, 2015.

[264] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[265] Chingiz Kabytayev, Todd J Green, Kaveh Khodjasteh, Michael J Biercuk, Lorenza Viola, and Kenneth R Brown. Robustness of composite pulses to time-dependent control noise. *Physical Review A*, 90(1):012316, 2014.

[266] CL Edmunds, C Hempel, RJ Harris, VM Frey, TM Stace, and MJ Biercuk. Dynamically corrected gates suppress spatio-temporal error correlations as measured by randomized benchmarking. *arXiv preprint arXiv:1909.10727*, 2019.

[267] Theodore Walter, Philipp Kurpiers, Simone Gasparinetti, Paul Magnard, Anton Potočnik, Yves Salathé, Marek Pechal, Mintu Mondal, Markus Oppliger, Christopher Eichler, et al. Rapid high-fidelity single-shot dispersive readout of superconducting qubits. *Physical Review Applied*, 7(5):054020, 2017.

[268] 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. *arXiv preprint arXiv:1905.11349*, 2019.

[269] Poulami Das, Swamit S Tannu, Prashant J Nair, and Moinuddin Qureshi. A case for multi-programming quantum computers. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 291–303. ACM, 2019.

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

[271] Andrew M Childs and Wim van Dam. Quantum algorithm for a generalized hidden shift problem. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1225–1232. Society for Industrial and Applied Mathematics, 2007.

[272] Don Coppersmith. An approximate fourier transform useful in quantum factoring. *arXiv preprint quant-ph/0201067*, 2002.

[273] Harper R Grimsley, Sophia E Economou, Edwin Barnes, and Nicholas J Mayhall. An adaptive variational algorithm for exact molecular simulations on a quantum computer. *Nature communications*, 10(1):1–9, 2019.

[274] Matthew Otten, Cristian L Cortes, and Stephen K Gray. Noise-resilient quantum dynamics using symmetry-preserving ansatzes. *arXiv preprint arXiv:1910.06284*, 2019.

[275] Swamit S Tannu and Moinuddin Qureshi. Ensemble of diverse mappings: Improving reliability of quantum computers by orchestrating dissimilar mistakes. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 253–265. ACM, 2019.

[276] Swamit S Tannu and Moinuddin K Qureshi. Mitigating measurement errors in quantum computers by exploiting state-dependent bias. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 279–290. ACM, 2019.

[277] Peter JJ O'Malley, Ryan Babbush, Ian D Kivlichan, Jonathan Romero, Jarrod R McClean, Rami Barends, Julian Kelly, Pedram Roushan, Andrew Tranter, Nan Ding, et al. Scalable quantum simulation of molecular energies. *Physical Review X*, 6(3): 031007, 2016.

[278] Deanna M Abrams, Nicolas Didier, Blake R Johnson, Marcus P da Silva, and Colm A Ryan. Implementation of the xy interaction family with calibration of a single pulse. *arXiv preprint arXiv:1912.04424*, 2019.

[279] Emanuel Knill, Dietrich Leibfried, Rolf Reichle, Joe Britton, R Brad Blakestad, John D Jost, Chris Langer, Roee Ozeri, Signe Seidelin, and David J Wineland. Randomized benchmarking of quantum gates. *Physical Review A*, 77(1):012307, 2008.

[280] Jinglei Cheng, Haoqing Deng, and Xuehai Qian. Accqoc: Accelerating quantum optimal control based pulse generation. *arXiv preprint arXiv:2003.00376*, 2020.

[281] Gian Giacomo Guerreschi. Scheduler of quantum circuits based on dynamical pattern improvement and its application to hardware design. *arXiv preprint arXiv:1912.00035*, 2019.

[282] Tzvetan S Metodi, Darshan D Thaker, Andrew W Cross, Frederic T Chong, and Isaac L Chuang. Scheduling physical operations in a quantum information processor. In *Quantum Information and Computation IV*, volume 6244, page 62440T. International Society for Optics and Photonics, 2006.

[283] 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.

[284] John L Hennessy and David A Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2011.

[285] Jeff Heckey, Shruti Patil, Ali JavadiAbhari, Adam Holmes, Daniel Kudrow, Kenneth R Brown, Diana Franklin, Frederic T Chong, and Margaret Martonosi. Compiler management of communication and parallelism for quantum computation. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 445–456, 2015.

[286] Gushu Li, Yufei Ding, and Yuan Xie. Towards efficient superconducting quantum processor architecture design. *arXiv preprint arXiv:1911.12879*, 2019.

[287] Prakash Murali, David C McKay, Margaret Martonosi, and Ali Javadi-Abhari. Software mitigation of crosstalk on noisy intermediate-scale quantum computers. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1001–1016, 2020.

[288] William K Wootters and Wojciech H Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, 1982.

[289] Peter Høyer and Robert Špalek. Quantum fan-out is powerful. *Theory of computing*, 1 (1):81–103, 2005.

[290] Yasuhiro Takahashi, Seiichiro Tani, and Noboru Kunihiro. Quantum addition circuits and unbounded fan-out. *arXiv preprint arXiv:0910.2530*, 2009.

[291] Yasuhiro Takahashi and Seiichiro Tani. Collapse of the hierarchy of constant-depth exact quantum circuits. *computational complexity*, 25(4):849–881, 2016.

[292] Frederic Green, Steven Homer, Cristopher Moore, and Christopher Pollett. Counting, fanout, and the complexity of quantum acc. *arXiv preprint quant-ph/0106017*, 2001.

[293] Yasuhiro Takahashi, Takeshi Yamazaki, and Kazuyuki Tanaka. Hardness of classically simulating quantum circuits with unbounded toffoli and fan-out gates. *Quantum Information & Computation*, 14(13-14):1149–1164, 2014.

[294] Daniel M Greenberger, Michael A Horne, and Anton Zeilinger. Going beyond bell's theorem. In *Bell's theorem, quantum theory and conceptions of the universe*, pages 69–72. Springer, 1989.

[295] Yao Lu, Shuaining Zhang, Kuan Zhang, Wentao Chen, Yangchao Shen, Jialiang Zhang, Jing-Ning Zhang, and Kihwan Kim. Global entangling gates on arbitrary ion qubits. *Nature*, 572(7769):363–367, 2019.

[296] Shruti Dogra, Kavita Dorai, and Arvind. Experimental construction of generic three-qubit states and their reconstruction from two-party reduced states on an nmr quantum information processor. *Physical Review A*, 91(2):022312, 2015.

[297] G Goldstein, P Cappellaro, JR Maze, JS Hodges, L Jiang, Anders Søndberg Sørensen, and MD Lukin. Environment-assisted precision measurement. *Physical review letters*, 106(14):140502, 2011.

[298] B Zeng, DL Zhou, and L You. Measuring the parity of an n-qubit state. *Physical review letters*, 95(11):110502, 2005.

[299] D Leibfried and David J Wineland. Efficient eigenvalue determination for arbitrary pauli products based on generalized spin-spin interactions. *Journal of Modern Optics*, 65(5-6):774–779, 2018.

[300] Koen Groenland, Freek Witteveen, Kareljan Schoutens, and Rene Gerritsma. Sequences of molmer-sorensen gates can implement controlled rotations using quantum signal processing techniques. *arXiv preprint arXiv:2001.05231*, 2020.

[301] SE Rasmussen, K Groenland, R Gerritsma, K Schoutens, and NT Zinner. Single-step implementation of high-fidelity n-bit toffoli gates. *Physical Review A*, 101(2):022308, 2020.

[302] Dongmin Yu, Yichun Gao, Weiping Zhang, Jinming Liu, and Jing Qian. Scalability and high-efficiency of an $(n+1)$-qubit toffoli gate sphere via blockaded rydberg atoms. *arXiv preprint arXiv:2001.04599*, 2020.

[303] 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.

[304] 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.

[305] Srinivasan Arunachalam, Vlad Gheorghiu, Tomas Jochym-O'Connor, Michele Mosca, and Priyaa Varshinee Srinivasan. On the robustness of bucket brigade quantum ram. *New Journal of Physics*, 17(12):123010, 2015.

[306] Olivia Di Matteo, Vlad Gheorghiu, and Michele Mosca. Fault-tolerant resource estimation of quantum random-access memories. *IEEE Transactions on Quantum Engineering*, 1:1–13, 2020.

[307] Ryan Babbush, Craig Gidney, Dominic W Berry, Nathan Wiebe, Jarrod McClean, Alexandru Paler, Austin Fowler, and Hartmut Neven. Encoding electronic spectra in quantum circuits with linear t complexity. *Physical Review X*, 8(4):041015, 2018.

[308] Maria Schuld and Francesco Petruccione. *Supervised learning with quantum computers*, volume 17. Springer, 2018.

[309] Maria Schuld and Nathan Killoran. Quantum machine learning in feature hilbert spaces. *Physical review letters*, 122(4):040504, 2019.

[310] Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209, 2019.

[311] Roohollah Ghobadi, Jaspreet S Oberoi, and Ehsan Zahedinejhad. The power of one qubit in machine learning. *arXiv preprint arXiv:1905.01390*, 2019.

[312] Kerstin Beer, Dmytro Bondarenko, Terry Farrelly, Tobias J Osborne, Robert Salzmann, and Ramona Wolf. Efficient learning for deep quantum neural networks. *arXiv preprint arXiv:1902.10445*, 2019.

[313] Sebastian Zaiser, Torsten Rendler, Ingmar Jakobi, Thomas Wolf, Sang-Yun Lee, Samuel Wagner, Ville Bergholm, Thomas Schulte-Herbrüggen, Philipp Neumann, and Jörg Wrachtrup. Enhancing quantum sensing sensitivity by a quantum memory. *Nature communications*, 7:12279, 2016.

[314] Juan Carlos Garcia-Escartin and Pedro Chamorro-Posada. Swap test and hong-ou-mandel effect are equivalent. *Physical Review A*, 87(5):052330, 2013.

[315] Maria Schuld, Mark Fingerhuth, and Francesco Petruccione. Implementing a distance-based classifier with a quantum interference circuit. *arXiv preprint arXiv:1703.10793*, 2017.

[316] Nathan Wiebe and Leonard Wossnig. Generative training of quantum boltzmann machines with hidden units. *arXiv preprint arXiv:1905.09902*, 2019.

[317] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, 2019.

[318] Gian Giacomo Guerreschi and Mikhail Smelyanskiy. Practical optimization for hybrid quantum-classical algorithms. *arXiv preprint arXiv:1701.01450*, 2017.

[319] Kosuke Mitarai and Keisuke Fujii. Methodology for replacing indirect measurements with direct measurements. *Physical Review Research*, 1(1):013006, 2019.

[320] Microsoft Q# Documentation. Estimategradient operation.

[321] Dorit Aharonov, Vaughan Jones, and Zeph Landau. A polynomial quantum algorithm for approximating the jones polynomial. *Algorithmica*, 55(3):395–421, 2009.

[322] Carlos Bravo-Prieto, Ryan LaRose, Marco Cerezo, Yigit Subasi, Lukasz Cincio, and Patrick J Coles. Variational quantum linear solver: A hybrid algorithm for linear systems. *arXiv preprint arXiv:1909.05820*, 2019.

[323] Xiaosi Xu, Jinzhao Sun, Suguru Endo, Ying Li, Simon C Benjamin, and Xiao Yuan. Variational algorithms for linear algebra. *arXiv preprint arXiv:1909.03898*, 2019.

[324] Hsin-Yuan Huang, Kishor Bharti, and Patrick Rebentrost. Near-term quantum algorithms for linear systems of equations. *arXiv preprint arXiv:1909.07344*, 2019.

[325] Stephen P Jordan. Fast quantum algorithms for approximating some irreducible representations of groups. *arXiv preprint arXiv:0811.0562*, 2008.

[326] Patrick J. Coles, S. Eidenbenz, S. Pakin, A. Adedoyin, John Ambrosiano, P. M. Anisimov, W. Casper, Gopinath Chennupati, C. Coffrin, Hristo Djidjev, D. Gunter, S. Karra, Nathan Lemons, Shizeng Lin, A. Lokhov, A. Malyzhenkov, D. Mascarenas, S. Mniszewski, B. Nadiga, D. O'Malley, D. Oyen, L. Prasad, R. Roberts, P. Romero, N. Santhi, N. Sinitsyn, P. Swart, Marc Vuffray, J. Wendelberger, B. Yoon, R. Zamora, and W. Zhu. Quantum algorithm implementations for beginners. *ArXiv*, abs/1804.03719, 2018.

[327] Sonika Johri, Damian S Steiger, and Matthias Troyer. Entanglement spectroscopy on a quantum computer. *Physical Review B*, 96(19):195136, 2017.

[328] Mikkel Kjaergaard, Maurice E. Schwartz, Andrew E. Greene, Gabriel O. Samach, Åke Bengtsson, Mark O'Keeffe, C. M. McNally, Jochen Braumuller, Dou Kyun Kim, Philip Krantz, Milad Marvian, A. Melville, Bethany M. Niedzielski, Yu-Suk Sung, Roni Winik, Josiah Yoder, Doug Rosenberg, Kristine Obenland, Stefan Lloyd, T. P. Orlando, Iman Marvian, Simon Gustavsson, and William David Oliver. A quantum instruction set implemented on a superconducting quantum processor. *arXiv: Quantum Physics*, 2020.

[329] Iman Marvian and Seth Lloyd. Universal quantum emulator. *arXiv preprint arXiv:1606.02734*, 2016.

[330] Nathan Wiebe, Ashish Kapoor, and Krysta M Svore. Quantum deep learning. *arXiv preprint arXiv:1412.3489*, 2014.

[331] Guillaume Verdon, Michael Broughton, and Jacob Biamonte. A quantum algorithm to train neural networks using low-depth circuits. *arXiv preprint arXiv:1712.05304*, 2017.

[332] Itai Arad and Zeph Landau. Quantum computation and the evaluation of tensor networks. *SIAM Journal on Computing*, 39(7):3089–3121, 2010.

[333] Scott Aaronson. Read the fine print. *Nature Physics*, 11(4):291–293, 2015.

[334] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical review letters*, 100(16):160501, 2008.

[335] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Architectures for a quantum random access memory. *Physical Review A*, 78(5):052310, 2008.

[336] Alexandru Paler, Oumarou Oumarou, and Robert Basmadjian. Constant depth bucket brigade quantum ram circuits without introducing ancillae. *arXiv preprint arXiv:2002.09340*, 2020.

[337] 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):646–650, 2017.

[338] Kenton R Brown, Andrew C Wilson, Yves Colombe, C Ospelkaus, Adam M Meier, E Knill, D Leibfried, and David J Wineland. Single-qubit-gate error below 10- 4 in a trapped ion. *Physical Review A*, 84(3):030303, 2011.

[339] J. P. Gaebler, T. R. Tan, Y. Lin, Y. Wan, R. Bowler, A. C. Keith, S. Glancy, K. Coakley, E. Knill, D. Leibfried, and D. J. Wineland. High-fidelity universal gate set for be 9+ ion qubits. *Physical review letters*, 117(6):060505, 2016.

[340] Klaus Mølmer and Anders Sørensen. Multiparticle entanglement of hot trapped ions. *Physical Review Letters*, 82(9):1835, 1999.

[341] Anders Sørensen and Klaus Mølmer. Quantum computation with ions in thermal motion. *Physical review letters*, 82(9):1971, 1999.

[342] Andre Bermudez, XiaoZhou Xu, Rinat Nigmatullin, Joe O'Gorman, Vlad Negnevitsky, Pam Schindler, Thomas Monz, Ulrich G. Poschinger, Christoph Hempel, Juanida Home, F. Schmidt-Kaler, Michael J. Biercuk, Rossella Blatt, Sherrin Benjamin, and M. Muller. Assessing the progress of trapped-ion processors towards fault-tolerant quantum computation. *Physical Review X*, 7:041061, 2017.

[343] Caroline Figgatt, Aaron Ostrander, Norbert M Linke, Kevin A Landsman, Daiwei Zhu, Dmitri Maslov, and Christopher Monroe. Parallel entangling operations on a universal ion-trap quantum computer. *Nature*, 572(7769):368–372, 2019.

[344] Raymond J Spiteri, Marina Schmidt, Joydip Ghosh, Ehsan Zahedinejad, and Barry C Sanders. Quantum control for high-fidelity multi-qubit gates. *New Journal of Physics*, 20(11):113009, 2018.

[345] Ye Wang, Stephen Crain, Chao Fang, Bichen Zhang, Shilin Huang, Qiyao Liang, Pak Hong Leung, Kenneth R Brown, and Jungsang Kim. High-fidelity two-qubit gates using a mems-based beam steering system for individual qubit addressing. *arXiv preprint arXiv:2003.12430*, 2020.

[346] 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.

[347] Dripto Debroy, Muyuan Li, Michael Newman, and Kenneth R Brown. Stabilizer slicing: Coherent error cancellations in ldpc codes. *arXiv preprint arXiv:1810.01040*, 2018.

[348] Patricia J Lee, Kathy-Anne Brickman, Louis Deslauriers, Paul C Haljan, Lu-Ming Duan, and Christopher Monroe. Phase control of trapped ion quantum gates. *Journal of Optics B: Quantum and Semiclassical Optics*, 7(10):S371, 2005.

[349] Sebastian Krinner, Simon Storz, Philipp Kurpiers, Paul Magnard, Johannes Heinsoo, Raphael Keller, Janis Luetolf, Christopher Eichler, and Andreas Wallraff. Engineering cryogenic setups for 100-qubit scale superconducting circuit systems. *EPJ Quantum Technology*, 6(1):2, 2019.

[350] M Müller, I Lesanovsky, H Weimer, HP Büchler, and P Zoller. Mesoscopic rydberg gate based on electromagnetically induced transparency. *Physical Review Letters*, 102 (17):170502, 2009.

[351] Nicholas Chancellor, Stefan Zohren, and Paul A Warburton. Circuit design for multi-body interactions in superconducting quantum annealing systems with applications to a scalable architecture. *npj Quantum Information*, 3(1):1–7, 2017.

[352] Samuel A Wilkinson and Michael J Hartmann. Many-body quantum circuits for quantum simulation and computing. *arXiv preprint arXiv:2003.08838*, 2020.

[353] Mohammadsadegh Khazali and Klaus Mølmer. Fast multiqubit gates by adiabatic evolution in interacting excited-state manifolds of rydberg atoms and superconducting circuits. *Physical Review X*, 10(2):021054, 2020.

[354] Chao Song, K. Xu, W. Liu, C. Yang, S. Zheng, Hui Deng, Qiwei Xie, K. Huang, Q. Guo, L. Zhang, P. Zhang, Da Xu, D. Zheng, X. Zhu, H. Wang, Y. Chen, C-Y Lu, S. Han, and J. Pan. 10-qubit entanglement and parallel logic operations with a superconducting circuit. *Physical review letters*, 119 18:180511, 2017.

[355] Kaiwen Gui, Teague Tomesh, Pranav Gokhale, Yunong Shi, Frederic T Chong, Margaret Martonosi, and Martin Suchara. Term grouping and travelling salesperson for digital quantum simulation. *arXiv preprint arXiv:2001.05983*, 2020.

[356] Markus Brink, Jerry M Chow, Jared Hertzberg, Easwar Magesan, and Sami Rosenblatt. Device challenges for near term superconducting quantum processors: frequency collisions. In *2018 IEEE International Electron Devices Meeting (IEDM)*, pages 6–1. IEEE, 2018.

[357] Hongye Yu, Frank Wilczek, and Biao Wu. Quantum algorithm for approximating maximum independent sets. *arXiv preprint arXiv:2005.13089*, 2020.

[358] Kristan Temme, Sergey Bravyi, and Jay M Gambetta. Error mitigation for short-depth quantum circuits. *Physical review letters*, 119(18):180509, 2017.