THE UNIVERSITY OF CHICAGO

COMPILATION, OPTIMIZATION AND VERIFICATION OF NEAR-TERM 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 PHYSICS

BY
YUNONG SHI

CHICAGO, ILLINOIS
DECEMBER 2020

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor Frederic T. Chong, without whom I would not be able to have such an incredible journey in the last three years. Fred not only influenced me with his sharp insights into science, problem solving skills but also with his reserved and easy-going demeanor, open-mindedness and positive attitude towards life. Fred always gives me the freedom to explore different areas of quantum computing and encourages me to think like a true researcher. Even though I am imperfect in a lot of ways, Fred is always patient with me and plans for the best for me.

This thesis would also not be possible without my wonderful collaborators. I would like to thank Andrew W. Cross, Christopher Chamberland and Ali Javadi for the help that I received from them during my visit of IBM T.J Watson research center. Andrew has opened the door of quantum error correction to me and I learned a lot from Andrew's incredible knowledge of every aspect of quantum computing. Chris, as a collaborator and a friend, showed me what characteristics makes one a true researcher. I would remember the sleepless nights we spent on calculations and the afternoons we went for basketball and climbing trees. Ali gave me tremendous support during my stay either by having fruitful discussions or providing resources for my projects, for which can't thank enough.

I would like to thank my collaborators from Columbia University, especially Professor Ronghui Gu and Runzhou Tao for the countless skype calls and hours spent on the CertiQ project.

I would like to thank my collaborators and friends from Schuster lab. I am indebted to Yao Lu and Nelson Leung for allowing me to "harass" them in the lab constantly and teaching me a lot about technical details about circuit QED. I would like to thank Professor David Schuster for being very supportive during our collaboration and during my faculty job search.

I would also like to thank my lab mates from EPiQC, including Jonathan Baker, Casey Duckering, Yonghshan Ding, Pranav Gokhale, Adam Holmes, Reza Jokar, Kate Smith, Ryan Wu. Because of them, being in the office and conference trips become a lot more enjoyable. It was an amazing experience to be a part of the wonderful EQiPC team.

Most importantly, I would like thank my family, my mom, my dad, my wife Xinya and Isaac, without whom my life would not be complete.

ABSTRACT

Quantum computing has recently transitioned from a theoretical prediction to a nascent technology [11]. Nevertheless, there is still a gap between the computational ability and reliability of current QIP technologies and the requirements of the first useful quantum computing applications. It will require tremendous efforts and investment to solve the engineering and research challenges to close the gap. Because of the uncertainties and difficulties in relying on hardware breakthroughs, it will also be crucial to close the gap using higher-level quantum optimizations and software hardware co-design, which could maximally utilize noisy devices and potentially provide an accelerated pathway to real-world quantum computing applications. This thesis develops methods to explore the vast design space for highly optimized and reliable quantum computing architectures and software and focuses on the compilation, verification and error correction aspects of the quantum computing stack. Specifically, the three main chapters in this thesis discuss, respectively, a direct-to-pulse compiler, a protocol to fault-tolerantly prepare approximate GKP (Gottesman-Kitave-Preskill) qubit states and a verification framework that helps quantum programmers write provably correct compiler components through formal verification. Though touching many aspects of the stack and based on different technologies in physics and computer science, the majority of this thesis can be connected by a main theme — improving the efficiency of existing quantum computing stacks by breaking the abstraction between layers in the quantum computing stack.

CHAPTER 1

THESIS OVERVIEW

Quantum Computing (QC) is at the cusp of a computing revolution. However, building the first realistic quantum computer remains a great challenge that requires the collaborative efforts of device physics, engineering, architectural design and software integration. My research goal is to explore the vast design space for highly optimized and reliable quantum computing stacks. Current quantum toolchains follow a vertically layered design similar to its classical counterpart: algorithms - programming languages - instruction sets - technology mapping - error correction/mitigation - physical implementation. This layered design philosophy hides details in lower level from each layer, which allows software to be simpler and less complex. However, this approach misses opportunities for optimization. In this thesis, we develop new methods to improve the computation efficiency and reliability in different phases of quantum computation, mainly by exposing lower level details to higher abstraction levels [183]. These methods focus on problems in areas from high level software development and verification [185, 5, 82] to compilation [184, 65] to physical implementation of qubits [182].

1.1 Breaking Abstractions in the Quantum Computing Stack

Breaking the ISA abstraction by pulse-level compilation One central task of QC systems is quantum compilation, which involves decomposing a quantum function into instructions (quantum gates). It was established in the gate-based circuit model more than 20 years ago that such decomposition is possible for a universal gate set of 1- and 2-qubit gates; however, no decomposition algorithms in the last 20 years can saturate or even approach the theoretical lower bound in terms of gate count and circuit latency, except in some special cases. In addition, there is a mismatch between the gate set that circuit decomposition/construction algorithms are targeting and the operations directly supported by the hardware — translating between them adds even more overhead. While improving computing efficiency is always valuable, improving quantum comput-

ing efficiency is do-or-die: computation has to finish before qubit decoherence or the results will be worthless. Thus, improving the compilation process is one of the most, if not the most, crucial goals in near-term QC system design. However, restricted by the ISA abstraction, current efforts to optimize gate-based compilation in academia and in industry achieve limited improvement.

My research approach to quantum compilation optimization is to break the ISA abstraction and directly compile sub-circuits down to control pulses [184, 65]. In contrast to the current gate-based compilation, the framework developed in [184] aggregates 1- and 2-qubit gates into larger operations. The compiler then finds commutative operations that allow for much more efficient schedules of control pulses (In fact, it is the first quantum compiler that utilizes the commutation relations of quantum mechanics in gate scheduling). It next uses GRAdient-Pulse-Engineering (GRAPE) method in Quantum Optimal Control (QOC) on the aggregates to produce a set of control pulses optimized for the the under-lying physical architecture. Leveraging QOC theory, this compilation method achieves 2-10X improvement in circuit latency for near-term quantum algorithms. In the follow-up work [65], it further considered the optimization of compilation time for near-term variational algorithms and achieves 100X speedup in compilation latency. In addition to the speedup, this line of work is the first realization that there is a huge design space for quantum compilers beyond the gate-based compilation and offers an architectural choice that supports more complex QC applications (through lower latency, and thus much lower error rates). In Chapter 3, we discuss in details about this compilation methodology.

Breaking the qubit abstraction by Bosonic qubit encodings In current QC stacks, the details of qubit implementations are often stripped away and only the (superpositions of) 0 and 1 information are retained. However, comparing to classical bits, quantum bits are far more fragile and short-lived. Thus, it's appealing to extract the rich error information provided by the physical qubit implementations and perform active error correction/mitigation. Bosonic codes like the Gottesman-Kitaev-Preskill (GKP) qubit encoding provide such architectures that enable upper level software to monitor the correctness of qubit states without destroying them. Based on

Continuous-Variable (CV) QC system, these Bosonic codes can be realized in superconducting cavities, offering a promising alternative to current superconducting qubits, which suffers from leakage errors, short coherence time and erroneous gate operations. In addition to its error correcting capability, the GKP encoding is advantageous in eliminating leakage errors, providing longer qubit lifetime (thanks to the long cavity coherence time) and supporting robust gate operations for universal computation. The main difficulty of information processing with the GKP code is to prepare the highly non-classical code state. In Chapter 4 and [182], we gave the definition of fault-tolerance for the preparation of GKP codes and designed a protocol that fault-tolerantly prepares GKP states. We also gave detailed analysis for realistic errors including nonlinear dispersive shift and Kerr nonlinearity in the protocol. This work for the first time defined fault tolerance for a qubit-cavity system and is one of the first attempt to bridge the fault-tolerance community and the CV code community.

With deeper integration with the upper level error correction, we believe that bosonic qubit architectures like the GKP qubit will serve as strong alternatives to the existing qubit implementations like the transmon qubits.

1.2 Reliable Quantum Software by Formal Verification

The verification of quantum tool chains is a newly developed area and is of great importance for near-term quantum computing. [185] describes the design and the implementation of the CertiQ verification framework [185] for the widely-used IBM Qiskit quantum compiler. CertiQ is the first mostly-automated verification tool for a real-world commercial quantum compiler. There are several challenges that makes formal verification of Qiskit difficult. First, checking the equivalence of quantum circuits is generally intractable. To mitigate this problem, CertiQ introduces the calculus of quantum circuit equivalence such that circuit equivalence and the correctness of compiler transformation can be statically and efficiently reasoned about. Based on the calculus, we design, specify, and verify a library of functions that perform primitive circuit transformations that are proved to be semantics preserving. Compilation phases implemented with this library can be

easily verified using symbolic reasoning. The second challenge is that compiler implementations in community code submission can be complicated, making automated verification intractable due to state explosion. In CertiQ, we developed a novel way of combining symbolic execution and quantum circuit calculus to achieve high level of automation and scalable verification. We verified four compiler phases and seven transpiler pass implementations of Qiskit in four case studies. With these verified CertiQ implementations, we successfully identify three bugs in Qiskit, two of which are unique in quantum software.

CHAPTER 2

INTRODUCTION TO QUANTUM COMPUTING

2.1 Quantum Computing Basics

A *qubit* state is a unit vector in the 2-dimensional Hilbert space $\mathbb{C}^2$ and is often written in the computational basis as a superposition of the logical states $|0\rangle = [1,0]^T$ and $|1\rangle = [0,1]^T$ in Dirac notation, *i.e.*,

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

where $|\alpha|^2 + |\beta|^2 = 1$ and $\alpha, \beta \in \mathbb{C}$. More specifically, a general qubit state can be parameterized by three real angles $\psi, \theta, \gamma \in \mathbb{R}$,

$$|\psi\rangle = e^{i\gamma}\cos(\theta/2)\,|0\rangle + e^{i\gamma}e^{i\phi}\sin(\theta/2)\,|1\rangle$$

There are also multi-qubit states which are unit vectors in higher dimensional Hilbert space. We call these quantum states as *pure states*. We write the conjugate transpose (dual) of $|\psi\rangle$ as $\langle\psi|$. In contrast, *mixed states* are classical ensembles of quantum states, which cannot be represented by quantum superpositions of $|0\rangle$ and $|1\rangle$. Instead, we use *density operators* to describing both mixed states and pure states. The density operator in the form $\rho = \sum P_i\,|\psi_i\rangle\,\langle\psi_i|$ states that the qubit in ρ has P_i probability of being in pure state $|\psi_i\rangle$. For both pure and mixed states, we have the trace of their density matrices $\mathrm{Tr}(\rho) = 1$, meaning that the total probability is 1. Also, density matrices are Hermitian, i.e., $\rho = \rho^\dagger$, where $\rho^\dagger$ is the conjugate of ρ. The composition of both state vectors and density operator representation are by tensor product, i.e., $|\psi_{q_1,q_2}\rangle = |\psi_{q_1}\rangle \otimes |\psi_{q_2}\rangle$, thus quantum system of n qubits is in the space $\mathbb{C}^{2^n}$ and grows exponentially.

It's often useful to map qubit state vectors to points on a unit sphere, which is known as the Bloch sphere [13]. In this representation, qubit states are parameterized by their spherical coordinates (α, β) and quantum gates become fixed-axis rotations. Evidently, the Bloch sphere rep-

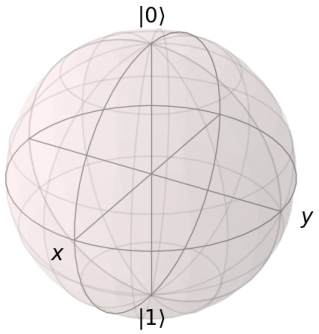
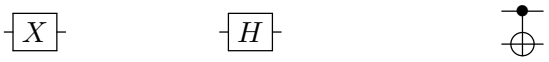


Figure 2.1: The Bloch sphere representation of a qubit state.

resentation is a projection of qubit states to a point on a 3d sphere, which can be seen trivially $\mathcal{B} : (\phi, \theta, \gamma) \rightarrow (\phi, \theta)$. For example, logical $|0\rangle$ with any global phase γ is mapped to the north pole and logical $|1\rangle$ with any global phase γ is mapped to the south pole and X gate becomes 180° rotation with regard to the x-axis. In the single qubit case, the Bloch sphere representation facilitates the combination of single qubit rotations. Usually, the implementation of the rotations on the Bloch sphere in quantum compilers is using Quaternion, which is also a method widely adapted in classical computer graphics.

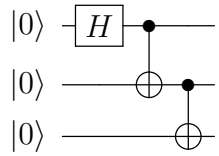


$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad CX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Figure 2.2: Circuit diagram symbols and matrix representations for several 1-qubit and 2-qubit gates.

Quantum gates are distance-preserving (or unitary) operations in this linear space. For example, the bit flip gate $X = [[0, 1], [1, 0]]$ maps $|0\rangle$ to $|1\rangle$ and $|1\rangle$ to $|0\rangle$, and Hadamard gate $H = \frac{\sqrt{2}}{2}[[1, 1], [1, -1]]$ maps $|0\rangle$ to $(|0\rangle + |1\rangle)/\sqrt{2}$. There are also multi-qubit gates such as CNOT, which fixes $|00\rangle$ and $|01\rangle$ while maps $|10\rangle$ and $|11\rangle$ to each other. We can see the first qubit in CNOT as the control qubit, i.e., if the first qubit is 0 then fix the state, otherwise flip the second qubit. Fig. 2.2

A quantum circuit is a sequence of quantum gates applied on an input quantum state to describe quantum algorithms. Quantum circuits are often represented symbolically in an Intermediate Representation(IR) (Figure 2.3). In different parts of the quantum compiler, different IR can be used. For example, the IBM Qiskit compiler [5] uses the DAG (Directed Acyclic Graph) implementation of quantum circuits during the compilation for flexible circuit manipulation and uses the OpenQASM IR [118] for input and output.




```
//GHZ circuit
OPENQASM 2.0;
include "qelib1.inc";

qreg q[3];
h q[0];
cx q[0],q[1];
cx q[1],q[2];
```

Figure 2.3: Circuit diagram and OPENQASM IR of a simple circuit.

2.2 Quantum Algorithms

We briefly describe several quantum algorithms that are advantageous comparing to existing classical algorithms in terms of computational complexity but requires fault-tolerant implementation of quantum computers, including Shor's algorithm [164] and Grover algorithm [71] and quantum simulation [49, 110] based on Quantum Phase Estimation (QPE) [95]. We also discuss several near-term quantum algorithms that could run on noisy quantum machines, including variational quantum eigensolver [117] and Quantum Approximate Optimization Algorithm (QAOA) [47].

2.2.1 Shor's algorithm

Shors algorithm was the main motivation of quantum computing research in its early day. Invented by Peter Shor in 1994, Shor's algorithm solves the integer factorization problem, which asks what are the prime factors of agiven integer N. Integer factorization problem has been known to be

hard for hundreds of years [163] and all known classical algorithms run with a time-complexity scales faster than or equal to $exp((lnN)^{\frac{1}{3}}(lnlnN)^{\frac{2}{3}})$. The long history of research in integer factorization problem makes people believe that it is classically intractable and the belief is so firm that it forms the theoretical foundation of the widely used RSA protocol for cryptography. However, in 1994, Peter Shor showed that, with the access to a (fault-tolerant) quantum computer, Shors algorithm runs with a time-complexity polynomial of $logN$, which is nearly exponentially faster than the previously known classical algorithms. This result suggests that once fault-tolerant quantum computers are built [67, 68, 69, 2, 99, 100, 162], current cryptography system will no longer be safe.

Shor's algorithm consists of a classical part and a quantum part. The classical part first decomposes the problem of factorization to solving a series of sub-problems in which the only computationally non-trivial sub-problem is to find the period r of the following function of real number x,

$$f(x) = a^x \ mod \ N$$

where a is a random integer that $1 < a < N$. Then the problem of solving for r is given to the quantum subroutine. The quantum subroutine of Shor's algorithm is performed in four steps:

- Initialization. A quantum register with $log_2 Q$ qubits is used to store the output r, where Q is an exponential of 2 between N^2 and $2N^2$. First, all qubits in this register is set to $|0\rangle$. Then a Hadamard gate is applied to each qubit in this register so that the initial state is $\frac{1}{\sqrt{Q}} \sum_{x=0}^{Q-1} |x\rangle$.
- Modular exponentiation. This step is to encode the function $f(x)$ into the unitary we apply.

The unitary corresponds to $f(x)$ does the following,

$$U_f \left|x, 0^q\right\rangle = \left|x, f(x)\right\rangle$$
$$U_f \frac{1}{\sqrt{N}} \sum_{x=0}^{Q-1} \left|x, 0^q\right\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{Q-1} \left|x, f(x)\right\rangle$$

where the 0^q part in the quantum state represents the second quantum register for output. It can be shown that such a unitary could be implemented by a sequence of controlled-"modular exponentiation" quantum gates.

- Inverse quantum Fourier transform. The ability to perform quantum Fourier transform differentiates Shor's algorithm with classical algorithm. In this step, we could transform the representation to the "frequency" domain and the resulting state encodes the "spectrum" of the frequencies.

- Measurement. After quantum Fourier transform, an output (y, z) that encodes the information of r for classical post-processing, where y, z are the output of first and second quantum register, respectively.

Since quantum algorithms are inherently statistical, the above subroutine is repeated for certain amount of times and it can be proved that, after some classical post-processing, we can solve for r with very high probability.

2.2.2 Grover algorithm

Grover algorithm is one of the first examples of quantum algorithms that are provably faster than classical algorithms [72]. Suppose we have a database with N entries that has one distinct entry and we want to design an algorithm to find that entry. We assume any searching algorithms could access a database query function. Generally, with each element represented by a $log_2 N$ bit binary string, the query function can be written as,

$$f_\omega(x) = \begin{cases} -1, & \text{if } x = \omega \\ 1, & \text{else} \end{cases} \tag{2.1}$$

The ω is the distinct member of the N discrete entries. It's easy to show that, classically, the query complexity is $O(N)$ since, with no priori knowledge, we have to query all entries in the database to find ω in the worst case. However, counter-intuitively, Grover's algorithm can be used for searching this unique ω in the database with the number of queries scaling as $\sqrt{N}$ [72]. If we want to understand it classically, this means that we don't even have to traverse all entries even in the worst case.

In Grover's algorithm, the database entries are encoded in quantum states. Also, the query function is given as a unitary quantum "oracle" U_q so that a quantum computer can have access. The algorithms starts with the superposition of all entries,

$$|\psi\rangle = v_0 |0\rangle + ... + v_{N-1} |N-1\rangle$$

where $v_i = \frac{1}{\sqrt{N}}$.

Then we define the Grover diffusion operator,

$$U_d = 1 - 2 |\omega\rangle \langle\omega|$$

We apply U_q and U_d alternately for $r(N)$ times. This procedure is called amplitude amplification as it can be geometrically interpreted as the the state vector is concentrated on the projection of ψ on $|\omega\rangle$. At last, we perform measurements to retrieve ω. It can be proven that $r(N) \sim O(\sqrt{N})$, the measured bitstring is arbitrarily close to N as $N \to \infty$.

2.2.3 *Quantum simulation*

The concept of quantum computers was first envisioned by Richard P. Feynman when considering the problem of simulating quantum systems. It's reasonable to believe that only computational tools that are intrinsically quantum mechanical can store and perform computation in the huge Hilbert space of realistic quantum systems.

The first concrete proposal of using quantum computers to simulate quantum systems was by Seth Lloyd in 1996 [110, 131]. The problem of simulating quantum systems comes down to reproduce the evolution of its Hamiltonian:

$$|\psi(t)\rangle = e^{-i\hat{H}t/\hbar} |\psi(0)\rangle$$

Classical simulation of the propagator $U(t)$ is limited by the memory usage when trying to store a potentially huge Hamiltonian. Memory usage is not a problem for quantum simulation, however, it has two new problems: first, qubits are bosonic, how to translate fermionic interaction to qubit interaction could be tricky; the second problem is how to discretize the Hamiltonian evolution so that it can be reproduced by the application of discrete quantum gates.

It turns out that the first problem can be solved by using the Jordan-Wigner transformation. We can map fermionic operators $a, a^\dagger$ to Pauli operators directly,

$$a_i \to \mathbb{I}^{\otimes i-1} \otimes \hat{\sigma}^+ \otimes \mathbb{I}^{\otimes N-i}$$
$$a_i^\dagger \to \mathbb{I}^{\otimes i-1} \otimes \hat{\sigma}^- \otimes \mathbb{I}^{\otimes N-i}$$

where N is the total number of qubits (particles) in the system and $\sigma^- = |0\rangle\langle 1|, \sigma^+ = |1\rangle\langle 0|$.

The second problem can be solved by observing that Hamiltonians are usually sums of local terms and using the Trotter-Suzuki formula to approximate the sum. Generally, if we have a

Hamiltonian $\hat{H} = \sum_i \hat{H}_i$, then with the (first order) Trotter formula, the propagator $U(t)$ writes,

$$\begin{aligned} U &= e^{-i\sum_j \hat{H}_j t/\hbar} |\psi(0)\rangle \\ &\approx \Big(\prod_j e^{-i\hat{H}_j t/(N\hbar)} \Big)^N \end{aligned}$$

where N can be seen as the number of time slices in the discrete simulation.

After Jordan-Wigner transformation, the above formula are products of exponentials of local Pauli operators, the unitaries of which can be manually constructed by single qubit rotation gates sandwiched by symmetric Controlled-NOT ladders.

To directly read out properties of interest, the circuit for the propagator is usually used as a subroutine in the Quantum Phase Estimation (QPE) algorithm, in which the eigenvalue (system energy) can be extracted by using quantum Fourier transform.

2.2.4 Variational Quantum Eigensolver

Quantum simulation method discussed in the last section is not practical in near-term since to reach the precision required by QPE, a fault-tolerant quantum computer is needed, of which the implementation involves millions of qubits and quantum gates.

To reduce the resource requirement of QPE, Varitional Quantum Eigensolver (VQE) was proposed [117, 142]. VQE is based on the variational principle and linearity of quantum observables. Different from traditional quantum simulation, VQE starts with an ansatz, which is a subspace in the whole Hilbert space that the resulting quantum state resides. Usually, ansatz can be prepared by a parameterized quantum circuit then finding the desired resulting quantum state boils down to finding the corresponding circuit parameters, $i.e.$, the vector $\vec{\theta}$ such that $|\psi(\vec{\theta})\rangle$.

Because of the variational principle, if the prepared state is, let's say, the ground state, then the expectation value of the Hamiltonian should be minimized. Thus, changing the circuit parameters in any way will only increase the expectation value of the Hamiltonian. Based on this observation, VQE measures the expectation value of the Hamiltonian for the ansatz and classically optimizes

the circuit parameters to minimize the expectation value, $i.e.$, finding

$$\min_{\vec{\theta}} \langle \psi(\vec{\theta}) | \, H \, | \psi(\vec{\theta}) \rangle$$

To summarize, VQE repeatedly performs the following steps until no further improvement can be found:

- Ansatz preparation
- Measurement of the expectation of the Hamiltonian.
- Improve the circuit parameters in the ansatz preparation using a classical computer.

2.2.5 Quantum Approximation Optimization Algorithm

Quantum Approximation Optimization Algorithm (QAOA) [47] is inspired by the Quantum Adiabatic Computing (QAC) model [3, 48], which encodes an optimization problem into Hamiltonian evolution.

To introduce QAOA, we describe QAC first. QAC performs computation in the following steps:

- Objective function encoding. The simplest example of objective function encoding is the Boolean satisfiability problem. Suppose we have n Boolean clauses $C_1, ..., C_n$ for m variables, C_n can be conveniently defined as $C_n = \sum_\alpha |\alpha\rangle \langle\alpha|$ where α are bit string configurations that satisfies C_n. It's easy to see each C_n is a real Hermitian matrix. Then the objective function is simply $H_p = \sum_{i=1}^{n} C_i$.
- Initialization of quantum states and Hamiltonian. QAC chooses an initial quantum state $|\psi_0\rangle$ and initial Hamiltonian H_0 where $|\psi_0\rangle$ is the eigenstate of H_0 with the lowest eigenvalue. H_0 is usually chosen to be $H_0 = \sum_i \hat{\sigma^x}_i$ and initial state will become the uniform superposition of all states.

- Quantum simulation of the Hamiltonian $H = (1-T)H_0 + TH_p$, starting with $|\psi_0\rangle$. Because of adiabaticity, we can make sure that at time T, the resulting state must be the eigenstate of H_p with the lowest eigenvalue.

- Measurement of the output bit string.

QAOA makes several modification to QAC to accommodate the limitations of near-term devices. In the quantum simulation step, QAOA decreases the number of time slices N in the trotter approximation to constant (usually 1 or 2). To compensate the error introduced by limited trotter steps, QAOA makes the coefficient $1 - T$ and T in the Hamiltonian free parameters that can be controlled individually and learned classically. In this way, circuit depth and gate count can be greatly reduced.

2.3 Quantum Architectures

Currently, there are several leading hardware platforms for near-term quantum information processing. These Quantum Information Processing (QIP) platforms include superconducting (SC) QIP platform, trapped ion QIP platform, quantum dot QIP platform, etc. As examples, we briefly introduce SC QIP platform and trapped ion QIP platform. First, we describe the DiVincenzo Criteria for a QIP platform.

2.3.1 The DiVincenzo criteria

In early 2000, theoretical physicist DiVincenzo listed necessary conditions for building a quantum computer, which is later known as the DiVincezo criteria [42]. There are five terms in the list:

- A scalable physical system with well characterized qubits.

- The ability of intializing qubit states.

- Long decoherence time for all qubits.

- The ability to implement a universal set of quantum gates.
- The ability to measure qubit states.

In the rest of this section, we will discuss these five aspects of two popular QIP platforms.

2.3.2 Superconducting QIP platform

Superconducting (SC) QIP platform [103, 39, 59] for quantum computation is advantageous in several ways: because the condensation of wave function of paired electrons ($i.e.$, Cooper pairs) in superconductors, microscopic quantum behaviors become macroscopic in near-absolute temperature, which enables simple characterization and precise control; superconducting qubits often operate at radio frequency with typical dimensions of micrometers, this allows the use of the well-established integrated circuit technology on the platform, which further allows flexible qubit design and information processing [53, 51].

On the SC QIP platform, quantum information is encoded in SC circuitry. Take the example of the simplest SC LC (Inductor-Capacitor) circuit, because of the macroscopic behavior of Cooper pairs in the circuit, the equation of motion can be characterized by a pair of canonical variables: capacitor charge n and inductor flux ϕ. Without external force, the LC circuit behaves like a typical quantum harmonic oscillator with the following Hamiltonian:

$$H = \omega\hbar(a^\dagger a + \frac{1}{2})$$

And the charge and flux operators can be expressed as

$$\phi = \sqrt{\frac{\hbar Z}{2}}(a + a^\dagger)$$
$$n = i\sqrt{\frac{\hbar}{2Z}}(a^\dagger - a)$$

where $Z = \sqrt{L/C}$ is the characteristic impedance of the oscillator. To use such a LC circuit as a qubit, we can encode logical 0 state in the lowest eigenstate and logical 1 state in the first excited

state of the oscillator. However, to selectively address the excitations, a non-linear element such as the Josephson junction is often used to make the circuit anharmonic. Write ϕ as $\phi_0\theta$ ($\phi_0 = \hbar/(2e)$), the Hamiltonian of anharmonic oscillators with a Josephson junction becomes,

$$H = 4E_c n^2 - E_J cos(\theta)$$

where $E_J = \phi_0 I_c$ (called the Josephson energy) and I_c is the superconducting critical current, $E_c = e^2/(2C_J)$ is the charging energy. When $E_c \gg E_J$, the oscillator is in the charge qubit regime, and the Hamiltonian can be directly diagonalized. When $E_c \ll E_J$, we are in the "transmon" qubit regime. In both situation, with properly designed circuit parameters, LC circuits could be served as qubits for quantum computation. In the transmon regime, the LC circuits are more resistant to charge noise.

The control and readout of superconducting qubits are done by replicating the light-matter interaction between optical cavities and natural atoms in the cavity QED system, where laser beams are replaced with microwave control pulses. The flexibility of designing the pulse patterns provide a wealth of possible interactions for superconducting qubits.

2.3.3 Trapped ion QIP platform

Trapped ion qubits [29] use internal electronic states of ions to store quantum information. Generally, there are four types of trapped-ion qubits: hyperfine qubits and fine structure qubits, which utilize hyperfine states, and fine structure levels, respectively; Zeeman qubits, which use splitted energy levels from magnetic fields; optical qubits, which use states separated by optical transitions. Each information encoding has its advantages and disadvantages.

Trapped ion qubits are initialized, controlled and measured by manipulating the internal state of collective motional state of ions using laser beams. For example, trapped ion qubits can be initialized by the optical pumping technique and 2-qubit (and multiqubit) gates can be realized by controlling the shared motional modes of two or more ions via .

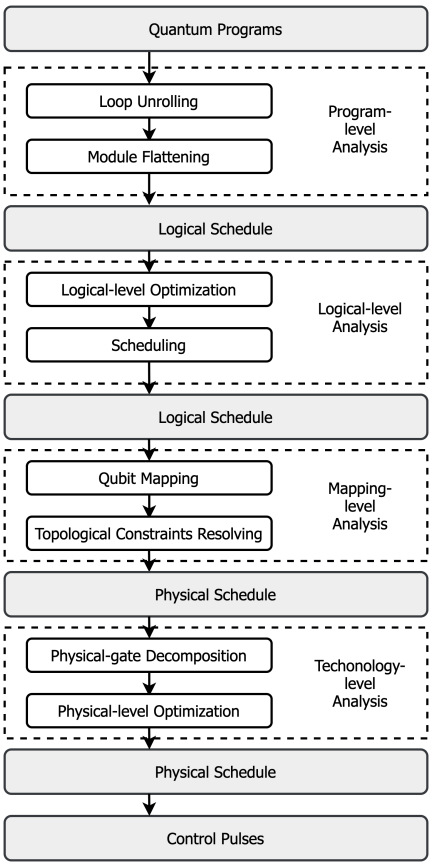


Figure 2.4: The workflow of the quantum computing software stack roughly followed by current programming environments ($e.g.$, Qiskit, Cirq, ScaffCC) based on the quantum circuit model.

Trapped ion QIP platform is advantageous in its long decoherence time and high fidelity quantum gates (though gates are slow) [37]. However, trapped ion systems do exhibit high leakage errors [192], ion loss and it is not yet known how to scale up its control system [122, 124, 17].

2.4 Software Stack of Near-term Quantum Computing

We are in the era of "Noisy Intermediate-Scale Quantum" (NISQ) technology [143], an early but necessary step towards fully fault-tolerant quantum computation. The NISQ software stack put more focus on error mitigation, compilation optimization than active error correction using redundant qubits [171, 161, 96, 100, 101]. Currently, major quantum programming environments, including Qiskit [5] by IBM, Cirq [1] by Google, PyQuil [169] by Rigetti and strawberry fields [172] by Xanadu, follow the quantum circuit model introduced earlier (Section 2.1). These program-

ming environments support users in configuring, compiling and running their quantum programs in an automated workflow and roughly follow a layered approach as illustrated in Fig. 2.4. In these environments, the compilation stack is divided into layers of subroutines that are built upon the abstraction provided by the next layer. This design philosophy is similar to that of its classical counterpart, which has slowly converged to this layered approach over many years to manage the increasing complexity that comes with the exponentially growing hardware resources. In each layer, burdensome hardware details are well encapsulated and hidden behind a clean interface, which offers a well-defined, manageable optimization task to solve [176, 87, 177, 26, 196, 74]. Thus, this layered approach provides great portability and modularity. For example, the Qiskit compiler supports both the superconducting QIP platform and the trapped ion QIP platform as the backend. In the Qiskit programming environment, these two backends share a unified, hardware-agnostic programming frontend even though the hardware characteristics and the qubit control methods of the two platforms are rather different. Fig. 2.5 illustrates the approach taken by current quantum software.

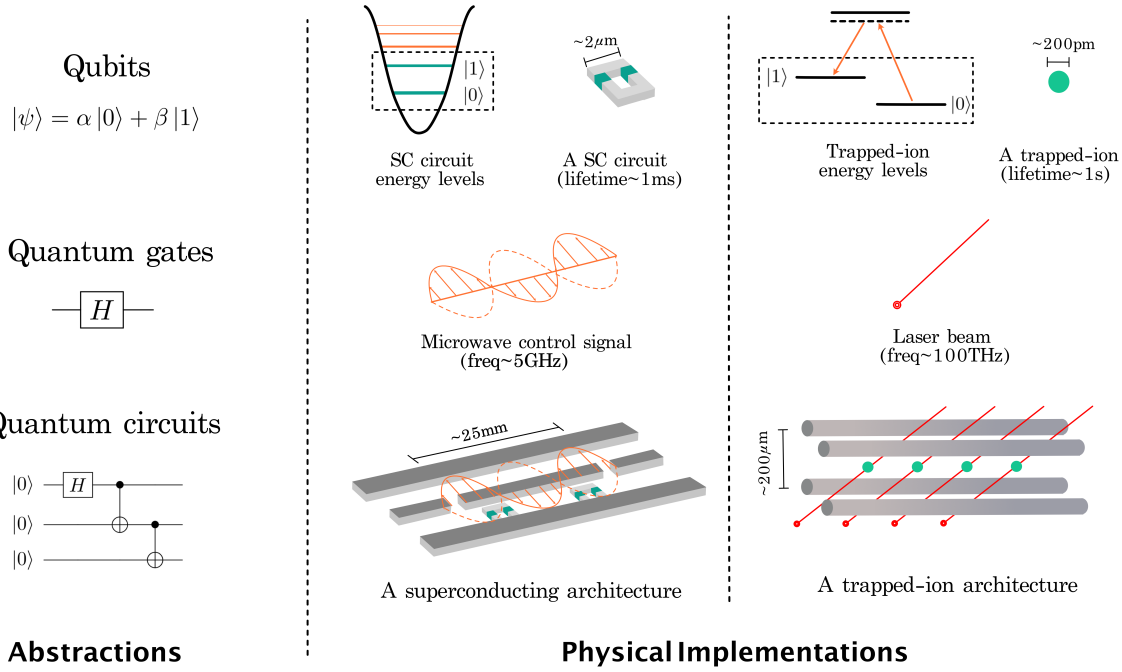


Figure 2.5: The same abstractions in the quantum computing stack on the logical level can be mapped to two different physical implementations.

CHAPTER 3

OPTIMIZED QUANTUM COMPILATION OF AGGREGATED INSTRUCTIONS

3.1 Quantum Compilation

In the early days of quantum computing, there are three reasons that people give high hope of the realization of quantum computing:

- The discovery of Shor's algorithm, which shows potential quantum advantage.
- The proof of threshold theorem, which states that we can suppress the error rate of a quantum computer to arbitrarily low (with layers of encoding).
- The discovery of Solovay Kitaev theorem, which states that quantum compilation is possible (given a universal set of quantum gates).

In this chapter, we focus on quantum compilation, which has been recognized as one of the central tasks in realizing practical quantum computation. Quantum compilation was first defined as the problem of synthesizing quantum circuits of 1- and 2-qubit gates for a given unitary matrix. The celebrated Solovay Kitaev theorem [94] states that such synthesis is always possible if a universal set of quantum gates is given. However, the problem of compiling the optimal circuits remains open.

Now the term of quantum compilation is used more broadly and almost all stages in Fig. 2.4 can be viewed as part of the quantum compilation process. These stages introduce more optimization problems that cannot be solved efficiently, for example, to map qubits onto hardware with limited qubit connectivity is an NP-hard problem. Also, we do not know how to transform from one basis of gates to another basis without introducing too much overhead in gate count.

In this chapter, we introduce a methodology that combines numerical methods from quantum control theory and classical scheduling algorithms to improve the depth of compiled circuits for near-term device. First, we briefly review some the basics of quantum compilation.

3.1.1 Solovay Kitaev theorem

We review the theoretical foundation of compiling quantum circuits. First, we give several important definitions.

Definition 1 (Operator norm) *The norm of an operator U is given by,*

$$\|U\| = sup \frac{\|U\,|\psi\rangle\|}{\||\psi\rangle\|}$$

The norm of quantum state $\||\psi\rangle\| = \sqrt{\langle\psi|\psi\rangle}$ is the Euclidean norm.

Definition 2 (Universal gate set) *A universal gate set $\mathcal{G}$ is a finite set of SU(4) whose free group $\langle G\rangle$ is dense in SU(4).*

where free group is defined in the usual sense in group theory. We can further define free group of length l (after cancellation), G^l be the group formed by l applications of elements in G. By dense, we mean that,

$$\forall u \in SU(4), \forall \epsilon > 0, \exists g \in \langle G\rangle : \|u - g\| < \epsilon$$

For clarity, we limited the definition of universal gate set to be 1- and 2-qubit gates. Clearly, there are gate sets include 3-qubit gates that can be used as a basis for compilation, $e.g.$, {Toffoli, H}. However, these gate sets can be further decomposed, thus we do not include them in the definition. It's possible to show that several commonly used gate sets are universal, $e.g.$, {CNOT, X, Y, Z, H, S, T}.

Further, we can define ε-net for ε that are not infinitesimal. For $A, G \in SU(4)$ we say a set S is a ε-net of A if

$$\forall u \in A, \exists s \in S : \|u - s\| < \varepsilon$$

There are two fundamental theorems about quantum compilation [94], both proposed by Alexei Kitaev.

Theorem 1 (Exact compilation) *Any unitaries can be exactly compiled to a quantum circuit with gates from SU(4).*

From Theorem 1, if we can implement all 1- and 2-qubit gates (which is true with current quantum control technology,see Section 3.2), we can exactly realize any quantum algorithms.

Theorem 2 (Approximate compilation, Solovay-Kitaev Theorem) *For any $d > 0$, there is a constant c such that for any universal gate set G and $\varepsilon > 0$ one can choose l = O($log^c(1/\varepsilon)$) so that G^l is an ε-net for SU(d).*

Solovay-Kitaev algorithm [94, 33] is most useful for transforming between quantum circuits with different gate sets, it tells us that the overhead of these transformations is bounded. Also, Theorem 1 and Theorem 2 to give a complete solution to the problem of compilation to a finite gate set.

We briefly sketch the proofs of the two theorems.

Proof sketch of Theorem 1 For Theorem 1, we start off by constructing a Toffoli gate using the following two unitaries,

$$A = \frac{1}{\sqrt{2}} \begin{pmatrix} -i & -1 \\ 1 & i \end{pmatrix}, B = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

Using A and B, a Toffoli can be constructed as in Fig. 3.1,

We can easily verify the circuit in Fig. 3.1 prepares a Toffoli gate: if the first two qubits are both $|1\rangle$, then $ABA^\dagger B^\dagger$ is applied, which flips the qubit state, however if either of qubit 1 and qubit 2 is in $|0\rangle$, the circuit will do nothing. This action fits the behavior of Toffoli gate.

Then we can prepare 2-controlled A and 2-controlled B gate with Toffoli and single qubit rotations. With the base case of 2-controlled A and 2-controlled B, we can we can recursively use

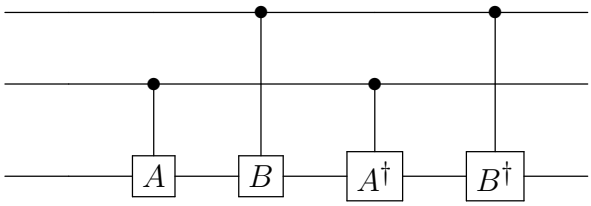


Figure 3.1: Construction of Toffoli gate

the above construction to prepare multi-controlled A and multi-controlled B gates by doubling the control bits and replace the controlled-A and controlled-B gates with the multi-controlled gates prepared in the last step.

With two applications of multi-controlled Toffoli and four applications of single qubit gates, we can prepare multi-controlled U for arbitrary single qubit gate U. Then with simple matrix multiplication, we can prove that arbitrary unitary matrices of dimension M can be represented by the product of $M(M-1)/2$ multi-controlled unitary gates. Then we finished the proof.

Proof sketch of Theorem 2 We provide the sketch of a non-constructive proof of Theorem 2 in the $d = 2$ case.

The most important ingredient in the proof is the "shrinking lemma".

Lemma 3 (Shrinking lemma) *There exist $\varepsilon', s \in \mathbb{R}$ such that for any gate set G and $\varepsilon < \varepsilon'$, statement 1 implies statement 2.*

Statement 1: G^l is an ϵ^2-net of an open ball of radius ε in SU(2).

Statement 2: G^{5l} is an $s\epsilon^3$-net of an open ball of radius $\sqrt{s\varepsilon^3}$ in SU(2).

The shrinking lemma implies that by increasing the number of quantum gates in the circuit, we can approximate a smaller "area" with better precision in the unitary space. The main idea of proving the shrinking lemma is to map unitaries in SU(2) to vectors in $\mathbb{R}^3$ and show the cross product of two unitaries can be approximated with increased precision.

Then by iteratively calling the shrinking lemma, we can prove the Solovay-Kitaev theorem. Similar results are also obtained by other researchers [41]. Empiricially, the constant c in the theorem has initially be found to be 3 by Kitaev. However, later, it was proven to be lower bounded

by 1 [76]. In 2012, an algorithm for the standardgate basis for fault tolerant quantum computing (Clifford + T) has been found to saturate this lower bound [98] based on basic number theory. Subsequently, improved methods has been proposed to reduce the number of ancillas [157, 154]. Recently, it is shown that optimal 2-qubit circuits of the fault-tolerant gate set can be constructed, too [62]. However, the optimal compilation of quantum circuits is still an open problem.

3.1.2 Other compilation methods

Most quantum algorithms are not described in the unitary representation, but are built constructively, thus do not require circuit synthesis using the compilation method mentioned in the last section. The resulting circuits are usually constructed by analytical methods. We take several algorithms as example.

Quantum Fourier transform Quantum Fourier transform (QFT) is widely used as a subroutine in quantuam algorithms such as Shor's algorithm and phase estimation. QFT in its product form is written as

$$|j_1 j_2 ... j_n\rangle \rightarrow \frac{(|0\rangle + e^{2\pi i 0.j_n}|1\rangle)(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle)(|0\rangle + e^{2\pi i 0.j_1 ... j_{n-1}j_n}|1\rangle)}{2^{n/2}}$$

We can define the following rotation gate,

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}$$

If we consider the state of the last qubit after QFT, $\frac{1}{2^{n/2}}(|0\rangle + e^{2\pi i 0.j_1 ... j_{n-1}j_n}|1\rangle)$. We can first look at the dependence of its initial state. It's easy to see that if we apply a Hadamard gate, we can replicate part of the above transformation that involves only the initial state of the last qubit itself

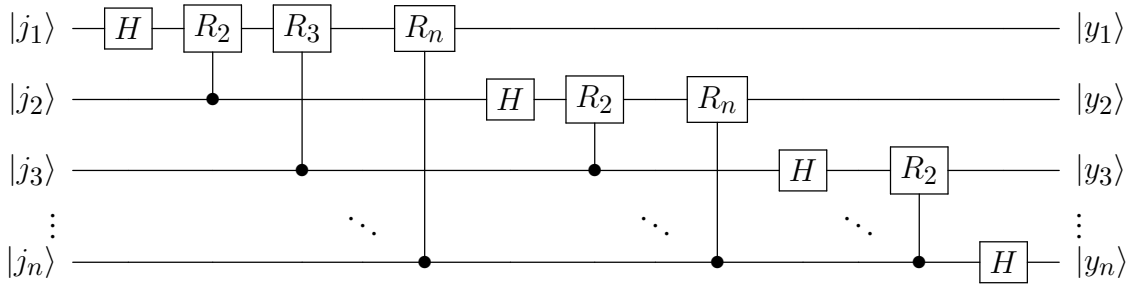


Figure 3.2: Quantum circuit for Quantum Fourier transform

($i.e.$, not care about other qubits yet), since,

$$H\left|j_1\right\rangle = \left|0\right\rangle + e^{2\pi 0.j_1}\left|1\right\rangle$$

Similarly, the dependence on the second qubit can be realized by a R_2 gate controlled by the second qubit:

$$CR_2\ H\left|j_1 j_2\right\rangle = \left(\left|0\right\rangle + e^{2\pi 0.j_1 j_2}\left|1\right\rangle\right)\left|j_2\right\rangle$$

With this observation, we can subsequently construct the circuit in Fig. 3.2 for QFT.

Quantum simulation Quantum simulation algorithm takes the description of an approximate Hamiltonian given by the Trotter formula. Generally, the description of an n-qubit quantum simulation have this form:

$$U = \prod_i U_i = \prod_i e^{\alpha_i P_i}$$

where $\alpha_i \in \mathbb{C}$ and P_i are Pauli operators. We can take a look at one of the exponentials,

$$U_i = e^{H_i}$$
$$H_i = \sigma_1^i \otimes \sigma_2^i \otimes ...\sigma_n^i$$

where $\sigma_j^i \in \{\sigma_x, \sigma_y, \sigma_z\}$.

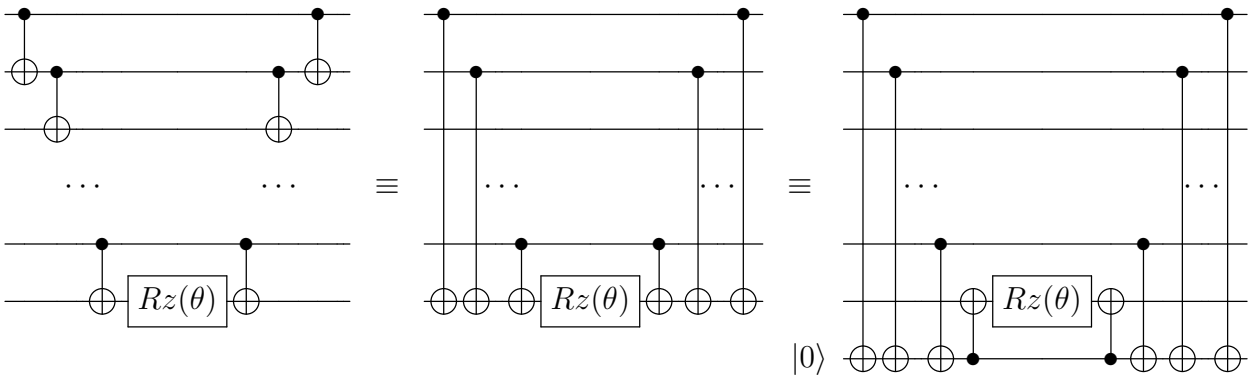


Figure 3.3: Circuits for the ZZ interaction in quantum simulation

Take the simplest case where $\sigma_j^i = \sigma_z$ for all j. It's simply a rotation on one of the qubit based on the parity of all qubits. Thus, the circuits in Fig. 3.3 can be constructed. There are three variants of the circuits which can be proved equivalent to each other. The circuit on the left is often referred as the "star" construction and the circuit in the middle is often referred as the "ladder" construction. The circuit on the right introduces an ancilla for better gate cancellation with other parts of the circuit.

3.2 Quantum Control

Before quantum information theorists formulate the problem of quantum compilation, physicists have already been thinking about the problem of controlling the evolution of their quantum systems in a best way. This is the problem concerned by the theory of quantum control.

In general, a finite dimensional control system has the following form,

$$\dot{\vec{x}} = f(t, \vec{x}, \vec{u})$$

where $\vec{x}$ represents the state of the system, usually is constraint on a submanifold of some larger space, f is a vector field on the manifold. $\vec{u} = \vec{u}(t)$ is the vector of control parameters.

For quantum control systems, the equation that describes their dynamics is the Schrodinger's

equation (or master equation if considering ensemables),

$$\frac{d}{dt}\vec{\psi} = -iH(u(t))\vec{\psi}$$

where H is the Hamiltonian, which is a Hermitian matrix and usually written as the sum of simpler parts,

$$H(u) = H_0 + \sum_k H_k u_k$$

It can be seen that quantum control systems can be modeled as bilinear systems as the right side of the Schrodinger's equation is a linear function of state $|\psi\rangle$ and it is an affine function of the control $\vec{u}$ from the equation above.

Sometimes it's more convenient to consider the unitary propagator,

$$\dot{U} = -iH(u)U$$

where for $U = U(t)$ is the propagator,*i.e.*, $\psi(t) = U(t)\psi(0)$.

3.2.1 Basic Lie algebra

We introduce some basic Lie algebra for the discussion of controllability later in this section.

Definition 3 (Lie algebra) *A vector space L over a field F, equipped an binary operation $L \times L \to L$, (denoted as $(x, y) \mapsto [x, y]$) is called a Lie algebra over F if the following properties are satisfied:*

- *The bracket operation is bilinear.*
- $[x, x] = 0$ *for all x in L.*
- $[x, [y, z]] + [y, [z, x]] + [z, [x, y]] = 0$ *for all x, y, z in L.*

It can be shown that the vector space of Hermitian matrices of dimension n is a Lie algebra. The most widely used example of Lie algebra in quantum information theory is (halves of) the Pauli matrices,

$$\sigma_x = \frac{1}{2}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_y = \frac{1}{2}\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \sigma_z = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

A *sub Lie algebra* A of Lie algebra L is a subset $A \subset L$ that is closed under the bracket operation.

Definition 4 (Lie group) *A Lie group is a group that is also a smooth manifold.*

The special unitary groups $SU(n)$ are Lie groups. Similarly, a sub Lie group H of a Lie group U is a sub-manifold H that also preserves and is closed under the group actions of U.

Theorem 4 (Correspondence between Lie algebra and Lie group) *There is a Lie algebra that is isomorphic to the tangent space of a Lie group at identity and a Lie algebra.*

For a Lie group of matrices, the elements in its corresponding Lie algebra can be obtained by differentiating curves at the identity. On the other hand, given a Lie algebra $\mathcal{L}$, its corresponding Lie group can be generated by,

$$e^{\mathcal{L}} = \{e^{l_1}e^{l_2}..., l_1, l_2, ... \in \mathcal{L}\}$$

3.2.2 Controllability

Now we discuss controllability of quantum systems. First, we define reachable sets. If we consider the states of the control system are unitary propagators, we define $R(t)$ to be the set of unitary matrices U_R such that there exists a control u with $U(u,t) = U_R$. Then the reachable set is the collection of $R(t)$ for all t,

Definition 5 (Reachable set) $R = \cup_{t\geq 0} R(t)$

It can be proved that reachable sets are Lie groups themselves.

Then, we define dynamic Lie algebra,

Definition 6 (dynamic Lie algebra) *Dynamic Lie algebra of a quantum control system is the vector space span*$_{u_k \in \vec{u}}\{-iH(u_k)\}$.

With knowing the dynamic Lie algebra of a control system, we can calculate the reachable set using the following theorem,

Theorem 5 *The reachable set is the Lie group corresponding to the dynamic Lie algebra* L, $i.e.$,

$$R = e^L$$

If the dimension of the reachable set R is the same with the dimension of the unitary space of the system, we say the quantum system is controllable. This also means if the dynamic Lie algebra can generate the corresponding Lie algebra of the whole unitary Lie group (usually $SU(n)$). As example, we can prove the following dynamic Lie algebra can generate the Lie algebra $\mathfrak{su}(4)$,

- $\{\sigma_x \otimes I, \sigma_y \otimes I, I \otimes \sigma_x, I \otimes \sigma_y, \sigma_x \otimes \sigma_x + \sigma_y \otimes \sigma_y\}$
- $\{\sigma_x \otimes I, \sigma_y \otimes I, I \otimes \sigma_x, I \otimes \sigma_y, \sigma_x \otimes \sigma_z\}$
- $\{\sigma_x \otimes I, \sigma_y \otimes I, I \otimes \sigma_x \otimes \sigma_x + \sigma_y \otimes \sigma_y + \sigma_z \otimes \sigma_z\}$

Thus, these are controllable quantum control systems. In fact, these control system are called XY interaction, ZZ interaction and Heisenberg interaction, respectively and are widely used in current QIP platforms to produce the physical $iSWAP$, CZ and $\sqrt{SWAP}$ gates (see Chapter A).

3.2.3 The KAK decomposition

KAK decomposition is a useful theorem about the structure of $SU(4)$.

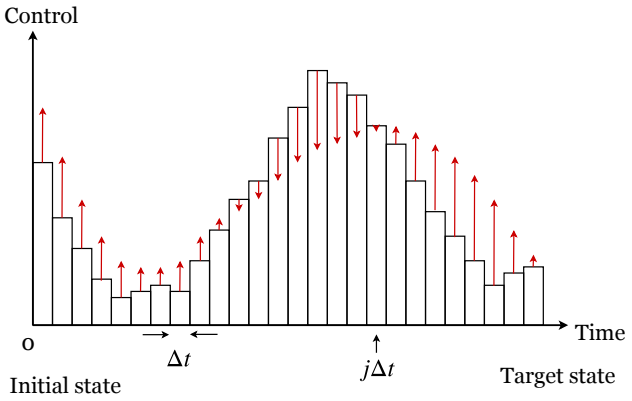


Figure 3.4: Quantum optimal control based on gradient descent, for a simplified single-pulse-type example. The black bars indicate the current iteration's proposed sequence of control pulse amplitudes by time interval, $\mu(j)$. The red arrows indicate the gradient of the output fidelity with respect to each $\mu(j)$. Thus, at the next iteration, each amplitude should be updated to $\mu(j)+\epsilon\frac{\partial L}{\partial \mu(j)}$, where L is the targeted loss function and ϵ is the adaptive step size.

Theorem 6 (KAK decomposition) *Any $U \in SU(4)$ can be written in the following form,*

$$U = (g_1 \otimes g_2)e^{i\vec{k}\cdot\vec{\sigma}}(g_3 \otimes g_4)$$

where for $i = 1, 2, 3, 4$, $g_i \in SU(2)$, $\vec{k} \in \mathbb{R}^3$ and $\sigma = (\sigma_x \otimes \sigma_x, \sigma_y \otimes \sigma_y, \sigma_z \otimes \sigma_z)$.

KAK provides important insights of the structure of $SU(4)$. Also, together with Theorem 1, KAK decomposition can be used to prove the possibility of quantum compilation.

3.2.4 Quantum optimal control using numerical methods

Quantum optimal control algorithms [61, 92, 35, 167, 46, 66, 133, 150, 141, 175, 194, 195, 138, 114, 137, 15, 40] find the optimal evolution path from a starting quantum state to a final quantum state, typically by performing gradient descent methods, such as the GRadient Ascent Pulse Engineering (GRAPE) [93, 34] algorithm. For a quantum system with a set of external control fields $u_1, \ldots, u_M$ that can be tuned in real-time, optimal control minimizes deviations from a target state by adjusting each control field u. In GRAPE, at every iteration the gradient of the target loss function (usually fidelity) with respect to a control field μ_k at time step j in the evolution can

be explicitly calculated by solving Schrödinger's equation. The algorithm will update the control field $\mu_k(j)$ in the direction of the gradient with adaptive step size ϵ [93, 34, 108] (Figure 3.4). With enough iterations, the converged control pulses are expected to drive the system from the initial state to the final state along an optimized path.

Gradient methods' running time and memory use grow exponentially with the size of the quantum system. In our work, we are able to numerically optimize quantum systems of up to 10 qubits with the GPU accelerated optimal control unit [108].

To implement this optimization, the time interval of interest is discretized into a large number N of sufficiently small time steps δt. Denoting intermediate times by $t_j = t_0 + j\,\delta t$, the Hamiltonian at time t_j takes on the form

$$H_j = \mathcal{H}_0 + \sum_{k=1}^{M} u_{k,j}\mathcal{H}_k. \tag{3.1}$$

The control fields subject to optimization now form a set $\{u_{k,j}\}$ of $d = M \cdot N$ real numbers.

The quantum evolution from the initial time $t = t_0$ to time t_j is described by a propagator K_j, decomposed according to

$$K_j = U_j U_{j-1} U_{j-2} \ldots U_1 U_0 \tag{3.2}$$

where

$$U_j = \exp(-iH_j\delta t) \tag{3.3}$$

is the propagator for the short time interval $[t_j, t_j + \delta t]$. (Here and in the following, we set $\hbar = 1$.) Evolution of a select initial state $|\Psi_0\rangle$ from $t = t_0$ to $t = t_j$ then takes the usual form

$$|\Psi_j\rangle = K_j|\Psi_0\rangle. \tag{3.4}$$

In the decomposition of K_j, each short-time propagator U_i can be evaluated exactly by matrix exponentiation or approximated by an appropriate series expansion.

3.3 Breaking the Abstraction of ISA using Aggregated Instructions

There are many indications that current quantum compilation stack (Fig. 2.4) is highly-inefficient. First, current circuit synthesis algorithms are far from saturating (or being closed to) the asymptotic lower bound in the general case [132, 94]. Also, the formulated circuit synthesis problem is based on the fundamental abstraction of quantum ISA and largely discussed in a hardware-agnostic settings in previous work but the underlying physical operations cannot be directly described by the logical level ISA (as shown in Fig. 2.5). The translation from the logical ISA to the operations directly supported by the hardware is typically done in an *ad-hoc* way. Thus, there is a mismatch between the expressive logical gates and the set of instructions that can be efficiently implemented on a real system. This mismatch significantly limits the efficiency of the current quantum computing stack thus the underlying quantum devices computing ability and wastes precious quantum coherence [97]. While improving the computing efficiency is always valuable, improving quantum computing efficiency is do-or-die: computation has to finish before qubit decoherence or the results will be worthless. Thus, improving the compilation process is one of the most, if not the most, crucial goals in near-term QC system design.

By identifying this mismatch and the fundamental limitation in the ISA abstraction, in [159, 64], we proposed a quantum compilation technique that optimizes across existing abstraction barriers to greatly reduce latency while still being practical for large numbers of qubits. Specifically, rather than limiting the compiler to use 1- and 2-qubit quantum instructions, our framework aggregates the instructions in the logical ISA into a customized set of instructions that correspond to optimized control pulses. We compare our methodology to the standard compilation workflow on several promising NISQ quantum applications and conclude that our compilation methodology has an average speedup of $5\times$ with a maximum speedup of $10\times$. We use the rest of this chapter to introduce this compilation methodology, starting with defining some basic concepts.

Gate	CNOT	SWAP	H	$R_z(\gamma)$	$R_x(\beta)$
Time (ns)	47.1	50.1	13.7	9.8	6.1
Gate	G_1	G_2	G_3	G_4	G_5
Time (ns)	54.9	13.7	42.0	31.4	6.1

Table 3.1: Instruction execution time for QAOA circuit in Fig. 3.5 (a). The pulse time for each gate in this table is optimized by an optimal control unit (see section 3.4.5). For the SWAP gate, we don't use the standard 3 alternating CNOTs implementation but optimize it individually.

3.4 Compilation Methodology

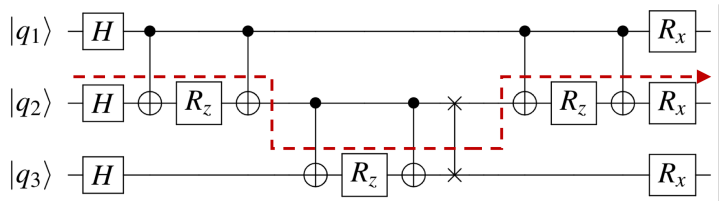
In this section, we demonstrate by example the advantage of our approach over standard gate-based compilation. Next we present our compilation methodology and introduce its end-to-end tool flow, including the frontend, backend, the optimal control unit, and verification procedure. In Section 3.5, we will detail the instruction aggregation algorithms.

3.4.1 An example of QAOA circuit

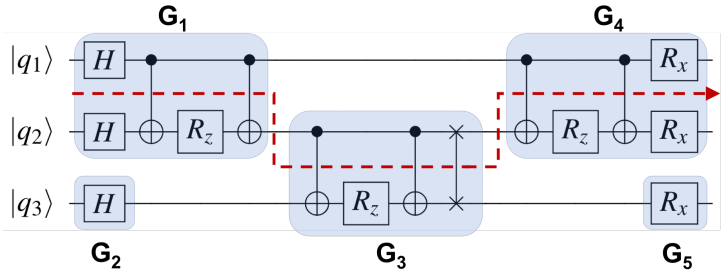
Fig. 3.5 (a) shows a quantum circuit that solves the MAXCUT problem for a triangle.[1] The circuit is decomposed into a standard gate set. This circuit (or variants of it up to single qubit gates) can be reproduced by most quantum software platforms, including ScaffCC, QISKit [31] and Pyquil [169]. We generate this circuit using ScaffCC. To keep our example small and realistic, we assume 1D nearest neighbor qubit connectivity and a underlying superconducting architecture. A SWAP gate is inserted to satisfy the qubit connectivity constraint. We choose to set the 1-qubit control field limit 5$\times$ the 2-qubit control field limit as a representative of real experimental settings [28]. The total execution time using gate-based compilation in Fig. 3.5 (a) is found by adding up the pulse time of each individual gate on the critical path of the circuit: $6T(CNOT) + T(SWAP) + T(H) + 3T(R_z) + T(R_x) = 381.9$ns using the numbers in Table 3.1.

In contrast, our compiler automatically generates the aggregated instruction set $G_1 - G_5$ as

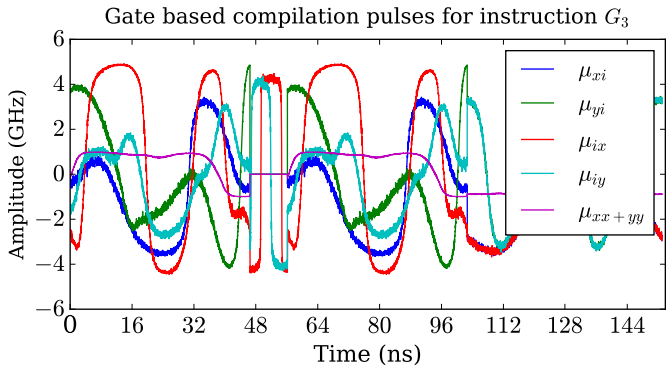
1. Specifically, the circuit implements the QAOA [47], one of the most promising near-term quantum algorithms, with angle parameters γ and β determined by variational methods [117] and set to 5.67 and 1.26.



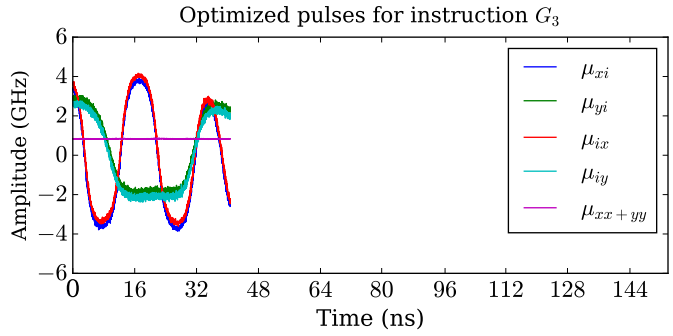
(a)



(b)



(c)



(d)

Figure 3.5: Example of a QAOA circuit demonstrating the difference between gate-based compilation and our compilation methodology. (a) Standard circuit (red arrow indicates the critical path). (b) Circuit with aggregated instructions. (c) Standard compilation pulses for G_3. (d) Aggregated compilation pulses for G_3. Each line represents the intensity of a control field. The pulse sequence in (d) is much shorter in duration and easier to implement than that of (c).

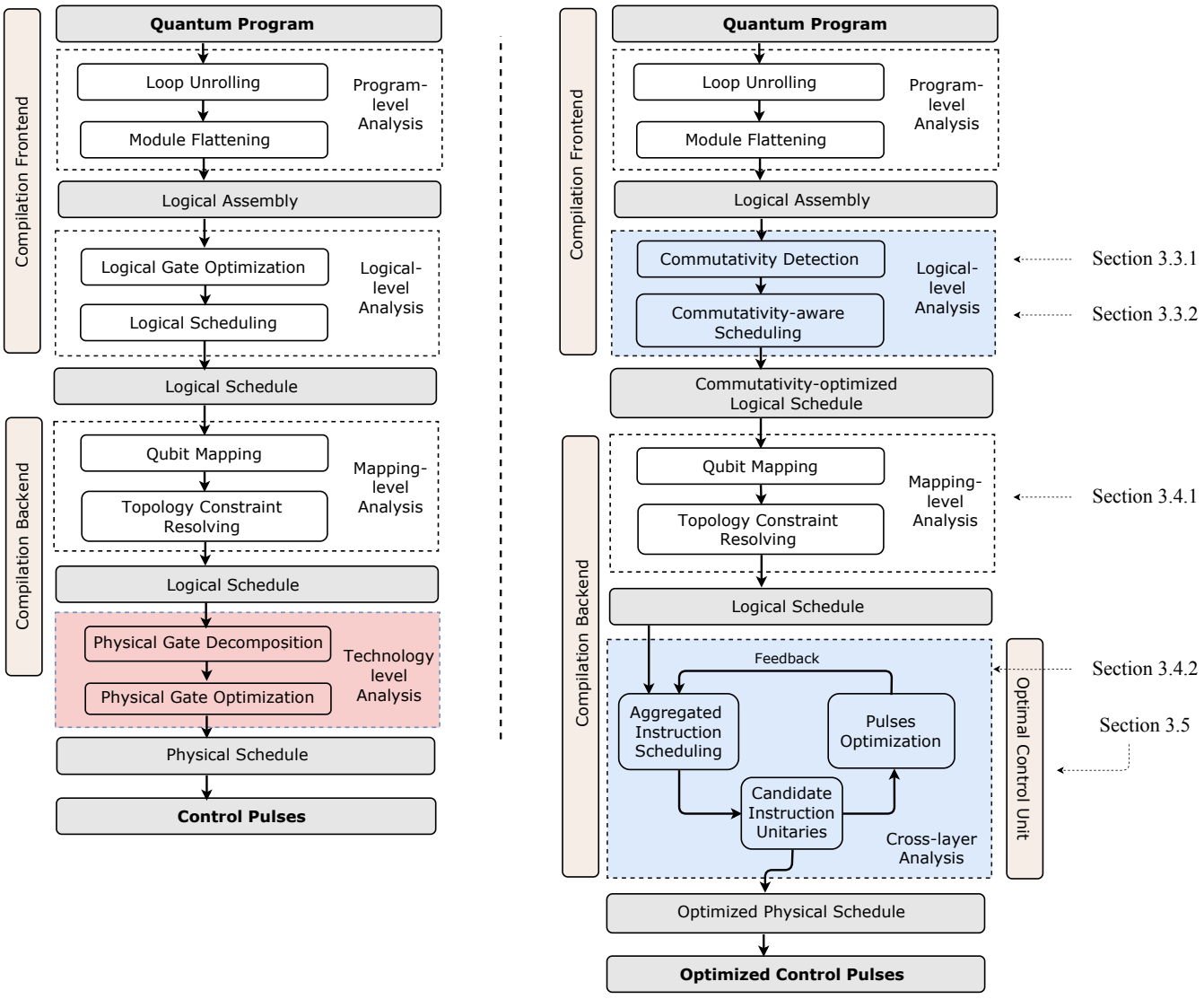
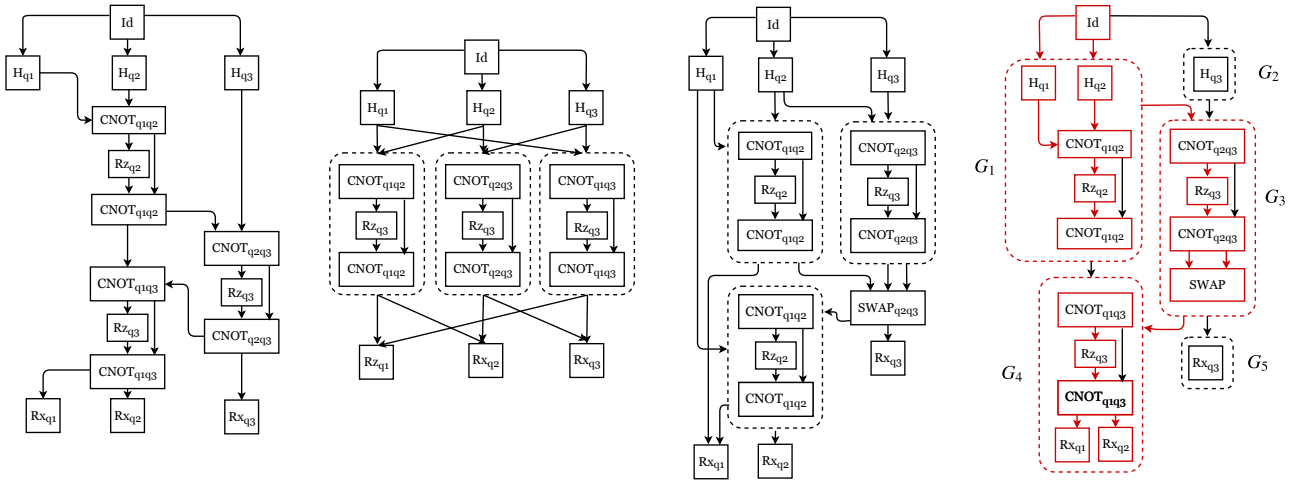


Figure 3.6: The comparison between standard gate-based compilation (left) and our compilation approach (right). The key differences are highlighted by the colored areas. In the first blue box, our compiler detects potential commutativity, which opens up opportunities for much more efficient scheduling. Then our logical scheduling takes advantage of commutativity for better parallelization. In the second blue box, by iterating with the optimal control unit, the instruction aggregation procedure breaks the well-encapsulated abstraction of 1- and 2-qubit logical gates and eliminates the physical gate layer (red box) that encodes only coarse-grained hardware information.



(a) Module flattening (b) Commutativity detection (c) Scheduling&mapping (d) Gate aggregation

Figure 3.7: The evolution of GDG for the circuit in Figure 3.5. In the compiler frontend, GDG in (a) is constructed for the flattened quantum program. By detecting commutative CNOT-Rz-CNOT instructions, the compiler transforms the GDG in (a) to GDG in (b) for more scheduling flexibility. Then, after scheduling and mapping, GDG has SWAP gates inserted and becomes GDG in (c). Finally, after the final aggregation, we arrive at the final GDG in (d), which is optimized both for parallelism and pulses generation. Each path in GDGs represents a qubit. The qubit name for each path is omitted in the figure for cleanness. The red paths in part (d) are the final critical paths.

indicated in Fig. 3.5 (b), and uses optimal control to produce minimal latency pulses for each. The pulse time for the circuit has critical path: $T(G_1) + T(G_3) + T(G_4) = 128.3$ns. In this example, our proposed aggregated instruction compilation reduces the pulse duration by about $2.97\times$ compared to standard gate-based compilation methods. Fig. 3.5 (c) and (d) compare the pulses for G_3 generated by gate-based compilation and generated by the optimal control unit.

3.4.2 Methodology overview

Figure 3.6 illustrates the key innovations in our proposed compilation scheme compared to standard gate-based compilation. Both approaches take a quantum program as input and proceed through a series of transformations to produce the control pulses that implement the computation on the physical qubits. In the traditional gate-based approach, the compiler first produces flattened quantum assembly codes, then generates a schedule of the logical instructions in the assembly codes. This schedule is later turned into a schedule of physical instructions by decomposing the logical instructions into physical instructions, which are converted into control pulses. We note that in the traditional gate-based approach, the physical properties of the underlying hardware are "localized" in each physical instruction. Compared to the traditional approach, our compilation process first converts assembly codes to a logical schedule that explores more commutativity by aggregating highly commutative instructions. Unlike traditional logical scheduling, our compiler aggregate highly commutative intermediate instructions in the assembly codes and generates a much more efficient logical schedule by re-arranging the new instructions. The logical schedule is then converted to a physical schedule after qubit mapping and SWAP gate insertion. At this point the compiler aggregates the final instructions and applies optimal control to the aggregated instructions. The goal is to find the optimal aggregation that produces the lowest-latency control pulses for the specified computation while considering aggregations that are small enough to be processed by the quantum optimal control unit. Output is an optimized physical schedule along with the corresponding optimized control pulses.

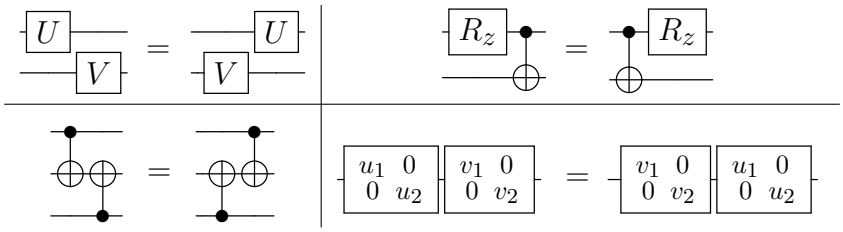


Table 3.2: Examples of gate commutation relations. Clockwise from top-left: gates acting on different qubits commute, controls commute with Z-axis rotations, gates with diagonal matrices commute, and CNOTs with disjoint controls commute.

3.4.3 Compilation frontend

The compiler frontend accepts quantum programs from the user, lowering high-level descriptions of quantum algorithms to a logical assembly that retains gate dependence relations. The compiler frontend performs program level analysis and preliminary logical level optimization, including loop unrolling, module flattening, commutativity detection, and logical level scheduling. The logical assembly output from the compiler frontend can be abstracted as a gate dependence graph (GDG) for each program.

Quantum GDG:

The main difference between a quantum GDG and a classical program dependence graph (PDG) is that quantum commutation rules apply in quantum GDG. More specifically, in a quantum GDG, consecutive commuting gates do not have parent-child relations [57] and can be scheduled in any order. Important commutation relations are depicted in Table 3.2.

In our compiler frontend, commutation relations between two gates A, B are resolved by explicitly checking the equality of unitary operators $\hat{A}\hat{B}$ and $\hat{B}\hat{A}$.

Figure 3.7 shows the GDG of the QAOA circuit in Figure 3.5 at different compilation stages. We insert an identity instruction as a virtual root for every GDG to connect instructions at depth 0. Because this virtual root is the identity instruction, it does not interfere with the computational result or latency. In our GDG, each path is labelled by a corresponding qubit name.

Commutativity detection:

Prior to commutativity detection, every consecutively scheduled pair of gates has a parent-child dependence. However, if a pair of gates commutes, then their relationship is a false dependence and the gates can be scheduled in either order. Our compilation technique relies heavily on the flexible scheduling of gates, so detecting commutativity and removing false dependencies in the GDG is critical for the rest of the compilation process.

In many near-term quantum applications, it is common for instructions *within* an instruction block to not commute, but for the full instruction blocks to commute with each other [47, 104]. As an example, in Fig. 3.5, the CNOT-Rz-CNOT structures commute with each other (these structures correspond to diagonal unitaries), but each CNOT and Rz in these structures does not commute. Thus in the GDG in Fig. 3.7 (b), after contracting the consecutive CNOT, Rz, CNOT instructions, the compiler is able to schedule new commuting CNOT-Rz-CNOT instructions in any order, while in the GDG in Fig. 3.7 (a), scheduling options are limited. This observation opens up opportunities for more efficient scheduling. In our design, the commutativity detection step achieves the goal of forming a highly commutative instruction set for the input quantum circuit. We detail the algorithm for commutativity detection in Section 3.5.

Commutativity-aware Logical Scheduling (CLS):

CLS uses commutativity, either detected from the last step or inherited from the original circuit, to extract more parallelism. With our GDG construction, it's natural to define the commutation group data structure on qubits. Each qubit maintains a list of commutation groups that, on that qubit, all the consecutive and commutative gates are in the same group. Two gates commute iff they are in the same commutation group on all the common qubits they share. This data structure facilitates more flexible scheduling and more optimization. For example, the two CNOTs in a CNOT-Rz-CNOT structure are in the same commutation group on the control qubit, but in different commutation groups on the target qubit. Then, with the commutation group data structure, we can correctly identify that any Rz gates on their control qubit can travel through these two CNOTs even

though these two CNOTs do not commute. Our CLS iterates the commutation groups on qubits in circuits. At each iteration, the CLS draws candidate gates to schedule from the first non-empty commutation groups on qubits, and schedules greedily. At each step, the candidate gates form a computational graph G_c with qubits as vertices and gates as edges (1-qubit gates are self-loops on a single vertex). The computational graph of candidate gates can conflict by sharing a qubit, in which case these gates cannot be scheduled simultaneously. The CLS then finds the maximal cardinality matching of G_c to resolve the conflicts. Fig. 3.8 illustrates an example of the maximal matching process. Algorithm 1 describes the CLS process.

Algorithm 1 CLS

Input:quantum GDG G_q, the list of commutation groups on qubits $\{com\_list[q_i] \mid q_i \in$ all qubits $\}$.
Output: logical schedule S.
Initialize current gates cg, next_time_point np, current commutation groups $\{com\_group[q_i] \mid q_i \in$ all qubits$\}$.
while cg not empty **do**
  candidate gates $ng = \{g$ can be scheduled at np— g in $cg\}$
  gates to be scheduled $gs =$ find_max_matching(ng)
  S+=gs; cg-=gs
  Update np
  for all $q_i \in$ all qubits **do**
    if $com\_group[q_i]$ empty **then**
      $com\_group[q_i] = com\_list[q_i].pop()$
    end if
  end for
  cg+=$\{g \mid g \in com\_group[q]$ for $q \in op(g)\}$
  Update com_group
end while
return S =0

Similar to previous work [57], our strategy is intended to maximize parallelism, and not to minimize the number of SWAP gates in the backend. Our motivation for this strategy in our compilation scheme is the finding that SWAP gates can be beneficial in reducing latency on superconducting architectures [155], so we don't aim to reduce the amount of SWAP gates. We also believe that a precise cost model that correctly discriminates the latency of each SWAP gate in circuits leads to more efficient scheduling strategies. We propose it as an exciting open problem.

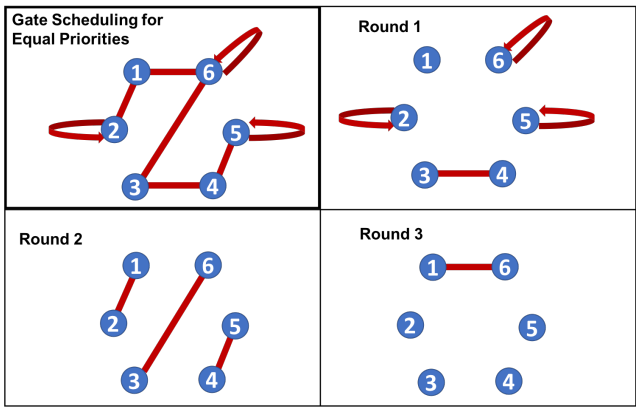


Figure 3.8: A computational graph with six qubits, all instructions have the same latency. The scheduler finds a maximal matching of non-adjacent edges and schedules them. The subsequent round repeats this process on the subgraph of remaining edges.

3.4.4 Compiler backend

The backend is responsible for mapping level optimization and final pulse generation. The backend executes the following steps: qubit mapping, topological constraint solving, and final instruction set aggregation.

Qubit mapping & topological constraint resolution:

Our logically-scheduled instructions do not account for the topological connectivity constraints of the underlying hardware. For the benchmarks presented in this paper (Table 3.3), we assume a rectangular-grid qubit topology with two-qubit operations only permitted between direct neighbors. This topology is representative of typical near-term superconducting quantum computers [145].

To conform to this topology, the logically-scheduled instructions are processed in two steps. First, we place frequently interacting qubits near each other by bisecting the qubit interaction graph along a cut with few crossing edges, computed by the METIS graph partitioning library [91]. As described in previous work [63, 81], this strategy is applied recursively on the partitions, yielding a heuristic mapping that reduces the distances of CNOT operations.

Once the initial mapping is generated, two-qubit operations between non-neighboring qubits

are prepended with a sequence of SWAP rearrangements that move the control and target qubits to be adjacent.

Instruction aggregation:

In this step, the compiler iterates with the optimal control unit to generate the circuit's final aggregated instructions. The optimal control unit optimizes each instruction individually. We describe how our instruction aggregation algorithm preserves parallelism in Section 3.5.3.

Finally, using the CLS from Section 3.4.3, the compiler schedules the circuit of aggregated instructions and sends the concatenated pulse sequences to the underlying hardware.

3.4.5 Optimal control unit

The optimal control unit in our compiler backend [108] provides optimized control pulses for each aggregated instruction. Our GPU accelerated quantum optimal control algorithm is based on automatic differentiation and the Tensorflow framework. Automatic differentiation allows users to specify advanced optimization criteria and easily incorporate them in pulse generation. These criteria include realistic experimental concerns like suppressing unwanted qubit levels, avoiding large voltage fluctuation, and most importantly, pulse latency.

The optimal control algorithm in our unit has been validated against real hardware and used in real experimental environments [77, 79].

3.4.6 Verification

Our framework uses the popular QuTip [84, 85] simulation backend to verify the quantum unitaries defined by the aggregated instructions and the resulting pulses generated by the optimal control unit. This verification procedure provides users confidence in the numerical accuracy of the results.

For our simulation (Section Section 3.6), we sample 10 aggregated instructions for each benchmark to verify that the control pulses of all instructions produce the correct unitary.

3.5 Instruction Aggregation

This section details the two algorithms for aggregating instructions in Section 3.4.3 and Section 3.4.4. We first discuss the allowed action space. Then we move onto our aggregation algorithms.

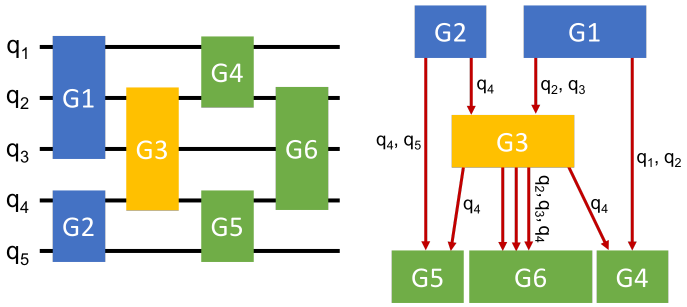


Figure 3.9: A circuit demonstrating the action space of instruction aggregation, with the corresponding GDG to the right. Gates in the same color group commute. G_3 can aggregate with any of the other gates. Only the action of aggregating G_3 and G_6 is monotonic in this circuit. All other aggregation pairs induce serialization upon the circuit by delaying a dependent aggregated instruction.

3.5.1 Action space for instruction aggregation

Here we define the allowed action space on GDG, where two instructions can aggregate if the following are true: 1. the two instructions overlap (share some common qubits); 2. one is the parent of the other on every qubit path they share or they are siblings; 3. If the two gates have parent-children relations, the parent (the children) either commutes with all gates in its commutation group on their common qubits or can be scheduled last (first) in the commutation group. In this way, we enforce the pulses inside an aggregated instruction to be continuous. In practice, we also limit the number of qubits in an aggregated instruction (instruction width) because of the scalability of the optimal control unit.

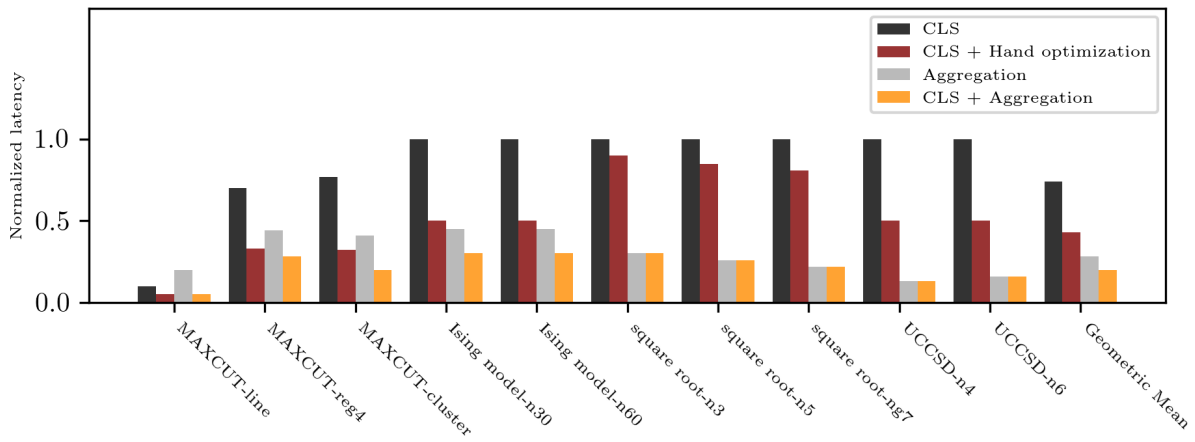


Figure 3.10: Normalized circuit latency of different strategies (ISA compilation is the baseline with latency 1.0).

Benchmark	Application Purpose	Qubits	Parallelism	Locality	Commutativity
MAXCUT-line	MAXCUT	20	Low	High	High
MAXCUT-reg4	MAXCUT	30	High	Medium	High
MAXCUT-cluster	MAXCUT	30	Medium	Low	High
Ising model	Solving Ising model	30	High	High	Medium
Ising model	Solving Ising model	60	High	High	Medium
square root-n3	Polynomial search	17	Low	High	Low
square root-n4	Polynomial search	30	Low	High	Low
square root-n5	Polynomial search	47	Low	High	Low
UCCSD-n4	UCCSD ansatz	4	Low	High	Low
UCCSD-n6	UCCSD ansatz	6	Low	Medium	Low

Table 3.3: Benchmarks

3.5.2 Diagonal unitaries aggregation for commutativity detection

To our knowledge, the most common commutative instructions are instructions representing diagonal unitaries because diagonal unitaries are used widely in decomposition methods of quantum chemistry applications [104] and near-term optimization algorithms [47]. To preserve parallelism, we only detect diagonal unitaries in blocks with a width of 2 qubits. To aggregate diagonal unitaries, we exhaustively search the action space defined in Section 3.5.1 within 2-qubit wide blocks(typically no longer than 10 gates).

3.5.3 Instruction aggregation

The main challenge of aggregating proper multi-qubit instructions is the conflict between parallelism and the need for larger instruction size for more speedup. Aggregating new instructions may potentially compromise parallelism. For example, in Figure 3.5, if G_5 is merged with G_3, then the circuit is serialized by the delay of G_4, which is dependent on G_3. To protect parallelism without querying optimal control unit too often, we make the following observation: for each aggregated instruction, the larger the instruction is, the more optimized the control pulses will be. Also, we notice that there is a set of allowed actions that will not delay critical paths even if the pulses in the new instruction are not optimized. We call these actions monotonic actions because in these actions, the reward of reducing circuit latency from aggregating a collection of instructions is strictly higher than aggregating a subset of the collection, as parallelism is not compromised. Monotonic actions can be checked by explicitly calculating the original circuit depth with the depth upon executing the action.

Our strategy is first to traverse the GDG. For each instruction in the GDG, we search the monotonic action set and keep the best action in a global table. After traversal on the GDG, we perform the global best action, and update the GDG and action table. We repeat until no more actions can be made. Then we update the latency of each aggregated instruction by querying the optimal control unit. This updated instruction latency could change the circuit structure and potentially create more monotonic actions, so we iteratively execute the above procedure until the GDG converges. For example, for the GDG in Figure 3.7 (c), after one iteration of instruction aggregation, we transform it to the circuit in Figure 3.7 (d). Figure 3.9 also illustrates an example of maintaining parallelism in the action space.

3.6 Evaluation

In this section, we present our simulation results. We first introduce our benchmark methodology. Then we present the main result — the latency between different compilation strategies. We con-

clude by analyzing the different factors that affect the final latency, including instruction width, parallelism, commutativity, and spatial locality.

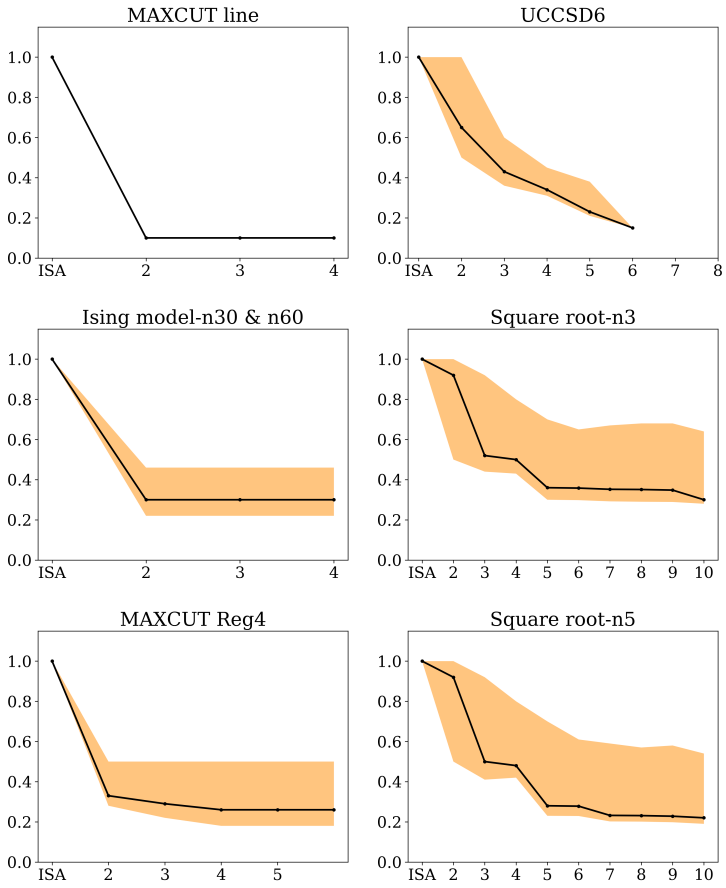


Figure 3.11: Allowed instruction width vs normalized latency in selected benchmarks. The black line is the normalized latency of the entire circuit. The upper (lower) edge of the filled area is the normalized latency of the instruction on the critical path that has the least (most) pulse optimization. The three applications in the left column are parallelized, either originally or after CLS. The three applications in the right column are serialized. Increasing the allowed instruction width will benefit serialized applications more.

3.6.1 Experimental setup

We perform our numerical study on superconducting architecture with XY interaction and set the control field limit of XY interaction to be $\mu_{max} = 0.02$GHz and single qubit rotation control field to be $5\mu_{max}$.By setting the control field strength to less than typical transmon anharmonicity, we model transmon operations with low leakage to high level states [28].

3.6.2 Benchmark methodology

We select several important classical-quantum hybrid algorithms and traditional quantum applications from the NISQ era as our benchmarks. The benchmarks are chosen to have different program characteristics that will affect the improvement from the aggregated instruction compilation. The complete list of benchmarks is shown in Table 3.3. The first three benchmarks are QAOA circuits solving MAXCUT problems [47]. These circuits are highly commutative. Ising model is a family of highly parallel circuits with limited commutativity. Square root circuits use Grover's algorithm [73] to find the square root of polynomials and they are very serialized. UCCSD stands for Unitary Coupled Cluster Single-Double ansatz [152] for the variational quantum eigensolver [117]. This ansatz is derived from the Jordan-Wigner or Bravyi-Kitaev transformations [104, 152] and is considered to be a machine unaware ansatz [152]. We include it to address that with optimal control, physics induced ansatzs can be made more hardware efficient on superconducting architectures and more competitive relative to machine-inspired ansatzs [88].

3.6.3 Latency

We present our main result in Figure 3.10. We compare four different strategies with unoptimized gate-based compilation. CLS refers to commutativity-aware scheduling (Section 3.4.3) Aggregation represents executing the instruction aggregation step (Section 3.5.3) without CLS; CLS + aggregation is self-explanatory. For hand optimization scheme, to the best of our knowledge, there are limited optimization methods documented for architectures with iSWAP gates ([155, 127]). Here hand optimization refers to mechanically applying the known methods ([155, 127]) with our best effort.

Across all 9 benchmarks, our compilation scheme achieves a geometric mean of $5.07\times$ pulse time reduction. CLS + hand optimization achieves a geometric mean of $2.338\times$ pulse time reduction. The program characteristics of each benchmark heavily affect the level of optimization by logical scheduling and aggregation, but our compilation scheme achieves better circuit latency than gate-based compilation with hand optimization for every benchmark studied here.

3.7 Discussion

3.7.1 Commutativity vs scheduling

In our study, the level of optimization from CLS scales with the commutativity of the circuit. In applications with little to no commutativity (like square-root, QFT and UCCSD), CLS has no effect as expected. In highly commutative circuits like MAXCUT circuits, CLS alone achieves up to $5\times$ circuit length reduction.

CLS also facilitates instruction aggregation. As shown in the Ising model-n15 example in Figure 3.10, CLS alone has no optimization, but CLS + aggregation arrives at a better optimization of $3.44\times$ circuit length reduction than $2.22\times$ for aggregation alone.

3.7.2 Parallelism vs instruction width

Figure 3.11 illustrates how circuit latency reduction scales with the allowed instruction width for several applications. For highly-parallel applications such as QAOA and the Ising model, the parallelism in the circuits places limits on the instruction width of aggregated instructions. Allowing a larger instruction width, therefore does not reduce latency. For serialized applications such as Square root and UCCSD, the latency reduction does not saturate until we reach the instruction width set by the scalability of the quantum optimal control.

In Figure 3.11, the lower bound of the yellow areas represents the largest latency reduction in an instruction on the critical path. In serialized applications, the total circuit latency reduction approaches this lower bound as instruction width increases. Thus, in these highly-serial applications, instructions with the largest latency reduction dominate the critical path, thus our

3.7.3 Spatial locality vs aggregation

To show how spatial locality affects the pulse optimization in our scheme, we compare the three instances of QAOA application in our benchmarks: MAXCUT-line, MAXCUT-reg4, MAXCUT-

cluster. After CLS, all of the three instances are highly paralleled and they have similar instruction sets that are composed of the CNOT-Rz-CNOT instruction and single qubit instructions. The main difference between the three instances is the spatial locality. The less spatially localized the instance is, the more SWAP gates must be inserted in the circuit.

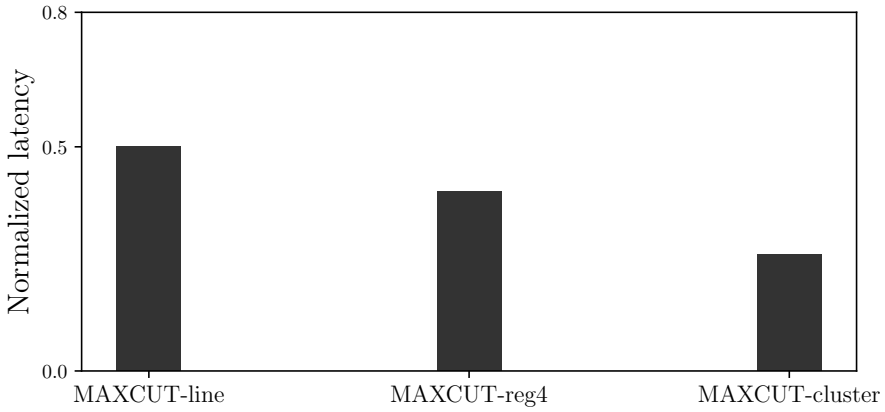


Figure 3.12: The normalized latency of 3 instances of QAOA applications in aggregated instruction compilation scheme. For each of these instances, the latency after performing CLS is set to be 1 as baseline. From left to right, the 3 instances have high, medium, and low spatial locality.

Figure 3.12 shows that the MAXCUT-cluster instance has the lowest latency and MAXCUT-line has the highest latency comparing to the normalized latency after performing CLS. For the same application, aggregated instruction compilation has larger improvements on circuits with low spatial locality.

3.7.4 Information encoding scheme vs pulse optimization

Information encoding schemes affect improvement due to pulse optimization. We evaluate the effect of the information encoding scheme on pulse optimization by comparing across spatially localized instances of our benchmark applications. QAOA applications encode the MAXCUT objective function directly onto the system Hamiltonian. In this simple encoding, inefficiency arises from the manual decomposition of diagonal unitaries generated by the objective Hamiltonian onto CNOT-Rz-CNOT instructions. In the spatially localized QAOA benchmark MAXCUT-line, hand optimization achieves about the same level of optimization as our compilation. UCCSD

applications map molecular structures by performing the Jordan-Wigner transformation [104] and then decomposing the corresponding diagonal unitaries onto CNOT-Rz-CNOT chains. In this more complicated information encoding scheme, our tool realizes $3.12\times$ greater circuit latency reduction than hand optimization in spatially localized instance UCCSD-n4. Our square root application involves reversible logic synthesis and quantum level decomposition, resulting in an encoding scheme that is more sophisticated than QAOA and UCCSD. In our Square root application, our tool realizes $3.68\times$ more circuit latency reduction than hand optimization.

From above observations, we see the trend that the more complicated the information encoding scheme is, the more advantageous our compilation is compared to hand optimization. This is expected, as simple hand optimization by replacing strategy is not efficient in finding the optimal path for complex quantum evolution with many degrees of freedom, especially when it involves sophisticated information encoding.

3.8 Related Work

Standard gate-based compilation is a well studied subject [80, 168, 31, 87, 52]. Practical techniques have been developed to improve the standard gate-based compilation from the reversible logic level down to the technology level, including studies of hand optimization, discrete [115, 116] and continuous [126] template matching, and rule-based rewriting [170, 120]. Template matching methods achieved impressive gate reduction on small and intermediate-scale circuits, though they are limited by having to manually search for new template rules for each specific gate library (for example, there is no library for iSWAP gates). Rule-based rewriting methods suffer from the huge search space of rewriting strategies and apply mainly to reversible level decomposition.

Because the abstraction of logical level instruction remains intact in the frontend of our compilation, our workflow is compatible with most of the optimization methods described above at logical gate level. However, these upper level optimization efforts might be canceled in the backend, *e.g.*, if template matching takes place within an aggregated instruction, it will cause no effect because the output unitary is the same.

Recent work has moved beyond standard ISA abstraction. Chuang et al [75] design a new Hamiltonian simulation method that reduces the problem of quantum simulation to optimal quantum control of single qubit rotations [75]. Google proposes a plan to construct random circuits to demonstrate quantum supremacy at the pulse level [128].

The use of optimal control to compile large-scale quantum circuits was first explored by Schulte-Herbrueggen et al. [156] in their restricted recursive-style complex quantum instructions where they report speedups up to 300%. The researchers, however, did not provide an instruction aggregation algorithm.

In this work, we provide a systematic and universal way to reduce the circuit latency the overcomes the disadvantages of previous works.

CHAPTER 4

FAULT TOLERANT PREPARATION OF APPROXIMATE GKP STATES

Fault-tolerant quantum computing will be essential for implementing large scale quantum algorithms that offer provable speed-ups over the best known classical algorithms. Currently there are many proposals for encoding qubits into error correcting codes in order to perform universal fault-tolerant quantum computation. Depending on the underlying physical architecture, some encoding schemes are more suitable than others.

One method proposed by Gottesman, Kitaev and Preskill is to encode a qubit into an oscillator such that small shift errors in both position and momentum can be corrected. Although some bosonic codes have been designed to correct realistic errors arising from noise models encountered in the experiment (e.g. photon loss), recently it has been shown that GKP codes have better error correction capabilities than such codes under the assumption of perfect encoding and decoding [4, 134, 179]. In addition, it has been shown how GKP codes can be concatenated with the toric code in order to achieve larger threshold values compared to toric codes with bare physical qubits [190, 54, 135]. Lastly, given a supply of GKP-encoded Pauli eigenstates, universal fault-tolerant quantum computation can be achieved using only Gaussian operations [12].

Given the above, it is clear that the fault-tolerant preparation of encoded GKP states is an important problem that needs to be addressed. Various proposals for preparing GKP states have been outlined [186, 60, 179, 8, 50, 173, 45, 43, 144]. However to our knowledge, no clear definitions for fault-tolerantly preparing GKP states using qubit-cavity couplings have been proposed. As such, without careful consideration, it is possible that a small number of faults lead to large uncorrectable shift errors. In this chapter, we discuss fault tolerant preparation of approximate GKP states. First, we introduce quantum error correction and GKP states. Inspired by [7], we propose new fault-tolerant definitions for preparing GKP states which tolerate small shift errors and a small number of faults occurring on ancilla qubits during the protocol. We then show how the phase estimation protocol proposed in this work satisfies our fault-tolerance criteria. In particular, the protocol is robust to a single fault occurring on the ancilla qubits in addition to shift errors of magnitude at

most 0.06. In order to be fault-tolerant, the protocol uses one flag qubit (and thus requires a total of two ancilla qubits) to prevent damping errors and imperfect implementations of the required gates from causing large shift errors. In addition, we provide an algorithm which only accepts a subset of all output states of phase estimation in order to prevent a single measurement readout error from causing large uncorrectable shift errors. We then proceed to show how our protocol can be implemented using circuit QED. We first analytically derive expressions describing the effects of the non-linear dispersive shift and Kerr non-linearity on the evolution of the cavity. We then numerically show that certain states output from the phase estimation protocol are robust to noise processes found in current 2D and 3D cavities since these can still correct small shift errors with high probability.

4.1 Quantum Harmonic Oscillator

Gottesman-Kitaev-Preskill (GKP) states (and bosonic codes in general) is constrcuted based on the quantum harmonic oscillator (QHO) model, which is also one of the most widely used and fundamental physical models in quantum mechanics. In this section, we introduce the QHO model. Similar to a classical pendulum, the QHO system can be characterized by a pair of canonical variables x and p, which are the position and momentum variable, respectively. The Hamiltonian of a QHO is given by,

$$H = \frac{kx^2}{2} + \frac{p^2}{2m} = \frac{1}{2}m\omega^2 x^2 + \frac{p^2}{2m},$$

where $\omega = \sqrt{\frac{k}{m}}$ is the angular frequency of. Differentiating H with respective to x and p, we can get,

$$\frac{\partial H}{\partial x} = m\omega^2 x = kx = -\dot{p}$$
$$\frac{\partial H}{\partial p} = \frac{p}{m} = \dot{x}$$

The above relations confirms that the variables x and p are canonical conjugate variables. In quantum mechanics, the canonical commutation relation holds between canonical conjugate variables. Thus, position x and momentum p have a commutator of $i\hbar$,

$$[x, p] = i\hbar.$$

With the above commutation relation, it's desirable to transform the Hamiltonian to a single term (plus constant terms). Thus, we can try the following transformation,

$$\begin{aligned} H &= \frac{m\omega^2}{2}(x - i\frac{1}{m\omega}p)(x + i\frac{1}{m\omega}p) - i\frac{1}{2}\omega[x, p] \\ &= \hbar\omega[\sqrt{\frac{m\omega}{2\hbar}}(x - i\frac{1}{m\omega}p)][\sqrt{\frac{m\omega}{2\hbar}}(x + i\frac{1}{m\omega}p)] + \frac{1}{2}\hbar\omega. \end{aligned}$$

To this point, it's convenient to define the operators a and corresponding adjoint $a^\dagger$ as

$$\begin{aligned} a &= \sqrt{\frac{m\omega}{2\hbar}}(x + i\frac{1}{m\omega}p) \\ a^\dagger &= \sqrt{\frac{m\omega}{2\hbar}}(x - i\frac{1}{m\omega}p) \end{aligned}$$

Using the canonical commutation relation of x and p,we can also derive the commutation relation of a and $a^\dagger$

$$\begin{aligned} [a, a^\dagger] &= 1 \\ [a^\dagger a, a^\dagger] &= a^\dagger \\ [a^\dagger a, a] &= -a \end{aligned}$$

Now the Hamiltonian equation can be re-written as,

$$H = \hbar\omega(a^\dagger a + \frac{1}{2})$$

After being simplified to a single non-constant term, it can be easily seen that the eigenstates of are is simply the eigenstates of $a^\dagger a$. Denote $|n\rangle$ be the nth eigenstate of $a^\dagger a$. If we we feed the state $a^\dagger |n\rangle$ into the Hamiltonian, we could find that it is also an eigenstate,

$$\begin{aligned}(a^\dagger a)a^\dagger |n\rangle &= (a^\dagger a^\dagger a + [a^\dagger a, a^\dagger]) |n\rangle \\ &= (a^\dagger a^\dagger a + a^\dagger) |n\rangle \\ &= (n+1)a^\dagger |n\rangle\end{aligned}$$

Similarly, $a |n\rangle$ is also an eigenstate of H.

$$(a^\dagger a)a |n\rangle = (n-1)a |n\rangle$$

Thus, given any eigenstate $|n\rangle$, one can use operators a and $a^\dagger$ to generate a ladder of eigenstates for the QHO Hamiltonian.

Next, we are interested in the norm of the eigenstate $a |n\rangle$, which can be calculated as follows,

$$\begin{aligned}n &= \langle n| (a^\dagger a |n\rangle) \\ &= (\langle n| a^\dagger)(a |n\rangle) \\ &= (a |n\rangle)^\dagger (a |n\rangle)\end{aligned}$$

Because $a |n\rangle$ is an eigenstate of $a^\dagger a$, we have $(a^\dagger a)a |n\rangle = (n-1)a |n\rangle$. Together with

$(a\,|n\rangle)^\dagger(a\,|n\rangle) = n$ calculated above, we can express $a\,|n\rangle$ as

$$a\,|n\rangle = \sqrt{n}\,|n-1\rangle\,,$$

Similarly, for $a^\dagger$, we could also get the following relation,

$$a^\dagger\,|n\rangle = \sqrt{n+1}\,|n+1\rangle\,.$$

With simple analysis of the ground state of the QHO model, we can prove that n must be integers and calculate the eigenvalues,

$$E_n = \hbar\omega(n+\frac{1}{2})$$

With above, we can explicitly calculate the wave function of the ground state in the x basis (or p basis),

$$\psi_0(x) = (\frac{m\omega}{\pi\hbar})^{\frac{1}{4}}\text{exp}(-\frac{m\omega}{2\pi\hbar}x^2)$$

We call the ground state of QHO the vacuum state. For the convenience of discussing GKP states, we also want to introduce the eigenstates of operator a (called coherent states) and the displacement operator.

Definition 7 (coherent state) *A coherent state $|\alpha\rangle$ is an eigenstate of the ladder operator a with eigenvalue $\alpha \in \mathbb{C} i.e., |\alpha\rangle = \alpha\,|\alpha\rangle$*

Definition 8 (displacement operator) *The displacement operator is defined as*

$$D(\alpha) = e^{\alpha a^\dagger - \alpha^* a}$$

It can be proved that displacement operators have the following properties,

- $D^\dagger(\alpha) = D^{-1}(\alpha) = D(-\alpha)$
- $D^\dagger(\alpha) a D(\alpha) = a + \alpha$
- $D^\dagger(\alpha) a^\dagger D(\alpha) = a^\dagger + a^*$
- $D(\alpha + \beta) = D(\alpha) D(\beta) e^{-i\mathrm{Im}(\alpha\beta^*)}$

As example, we prove the second and the fourth property.

Proof of the second property:

$$D^\dagger(\alpha) a D(\alpha) = e^{\alpha a - \alpha^* a^\dagger} a e^{\alpha a^\dagger - \alpha^* a} = a + [\alpha^* a - \alpha a^\dagger, a] =$$
$$= a + \alpha^*[a, a] - \alpha[a^\dagger, a] = a + \alpha$$

Q.E.D.

In the above, we used the Baker-Campbell-Hausdorff formula. Commutators above the first order vanish, thus leaving a simple form in the final expression.

Proof of the fourth property:

$$D(\alpha + \beta) = e^{\alpha a^\dagger - \alpha^* + \beta a^\dagger - \beta^* a} = e^{\alpha a^\dagger - \alpha^* a} e^{\beta a^\dagger - \beta^* a} e^{-\frac{1}{2}[\alpha a^\dagger - \alpha^*, \beta a^\dagger - \beta^* a]} -$$
$$D(\alpha) D(\beta) e^{-\frac{1}{2}(\alpha\beta^* - \alpha^*\beta)} = D(\alpha) D(\beta) e^{-i\mathrm{Im}(\alpha\beta^*)}$$

Q.E. D.

Sometimes, displacement operators are often called shift operators. In this thesis, we use them interchangeably. Next, we introduce a useful theorem for GKP states,

Theorem 7 *The coherent state $|\alpha\rangle$ is generated from vaccum $|0\rangle$ by the displacement operator $D(\alpha)$,*

$$|\alpha\rangle = D(\alpha) |0\rangle$$

Finally, we introduce the Wigner function, which can help understand GKP states in a visually clear way.

Definition 9 *The Wigner function of a wave function in the phase space of x and p is defined as,*

$$W(x,p) = \frac{1}{\pi\hbar}\int_{-\infty}^{\infty} \psi^*(x+y)\psi(x-y)e^{2piy/\hbar}dy$$

The definition of Wigner is inspired by the concept of joint probability in classical probability theory. As we can see in the next section, the characteristics of GKP states is most distinct when presented in the Wigner function.

4.2 Gottesman Kitaev Preskill State

With the QHO model introduced, we can now define a GKP state. We start with a perfect GKP state which exists only in theory, then move on to the definition of the physically realizable approximate GKP state.

4.2.1 Perfect GKP states

GKP states in the QHO model can be used to encode a discrete qubit, thus it can be seen as an error correction code to encode logical qubits. We first introduce the logical states of the GKP code. The perfect code words of the GKP codes are:

$$|\bar{0}\rangle = \sum_{s=-\infty}^{\infty} |q = 2s\sqrt{\pi}\rangle = \sum_{s=-\infty}^{\infty} |p = s\sqrt{\pi}\rangle$$

$$|\bar{1}\rangle = \sum_{s=-\infty}^{\infty} |q = (1+2s)\sqrt{\pi}\rangle = \sum_{s=-\infty}^{\infty} |p = (-1)^s s\sqrt{\pi}\rangle$$

$$|\bar{0}\rangle + |\bar{1}\rangle = \sum_{s=-\infty}^{\infty} |q = s\sqrt{\pi}\rangle = \sum_{s=-\infty}^{\infty} |p = 2s\sqrt{\pi}\rangle$$

$$|\bar{0}\rangle - |\bar{1}\rangle = \sum_{s=-\infty}^{\infty} |q = (-1)^s s\sqrt{\pi}\rangle = \sum_{s=-\infty}^{\infty} |p = (1+2s)\sqrt{\pi}\rangle$$

The Wigner function of logical states of the GKP code are illustrated in Fig. 4.1.

Some remarks:

- The states on the code space are the +1 eigenstates of q-shift operator $S_p = e^{-2\sqrt{\pi}ip}$ and p-shift operator $S_q = e^{2\sqrt{\pi}iq}$. This can be easily verified by shifting the δ peaks in the wavefunctions.

- To see the conversion of the wave function representation from the q basis to p basis, we just apply the Fourier transform.

$$
\begin{aligned}
|\bar{0}\rangle &= \sum_{s=-\infty}^{\infty} \delta(q = 2s\sqrt{\pi}) = \int \sum_{s=-\infty}^{\infty} e^{ipq}\delta(q = 2s\sqrt{\pi})dq = \sum_{s=-\infty}^{\infty} e^{2is\sqrt{\pi}p} \\
&= \sum_{s=-\infty}^{\infty} \delta(p = s\sqrt{\pi}) \\
|\bar{1}\rangle &= \sum_{s=-\infty}^{\infty} \delta(q = (2s+1)\sqrt{\pi}) = \int \sum_{s=-\infty}^{\infty} e^{ipq}\delta(q = (2s+1)\sqrt{\pi})dq = \sum_{s=-\infty}^{\infty} e^{(2s+1)i\sqrt{\pi}p} \\
&= e^{i\sqrt{\pi}p} \sum_{s=-\infty}^{\infty} \delta(p = s\sqrt{\pi}) = \sum_{s=-\infty}^{\infty} (-1)^s \delta(p = s\sqrt{\pi})
\end{aligned}
$$

  Here the normalization is omitted and we used the relation $\sum e^{-2is\sqrt{\pi}p} = \sum e^{-2is\sqrt{\pi}p} = \sum \delta(p = s\sqrt{\pi})$.

- In the original GKP proposal, the spacing between the δ functions can be tuned by a parameter α such that the spacing between the peaks in the q-space wave function is 2α and in the p-space is $\frac{\pi}{\alpha}$ in $|\bar{0}\rangle$(so the correctable errors are $\alpha/2$ and $\frac{\pi}{4\alpha}$). It's reasonable to set $\alpha = \sqrt{\pi}$ and let the correctable shift errors comparable in both space.

4.2.2 Approximated GKP states

Since the GKP states are not physical(energy $\to \infty$), they can only be approximateed in experiments. Usually the proposed approximation is to use a Gaussian wave(with width Δ) to replace

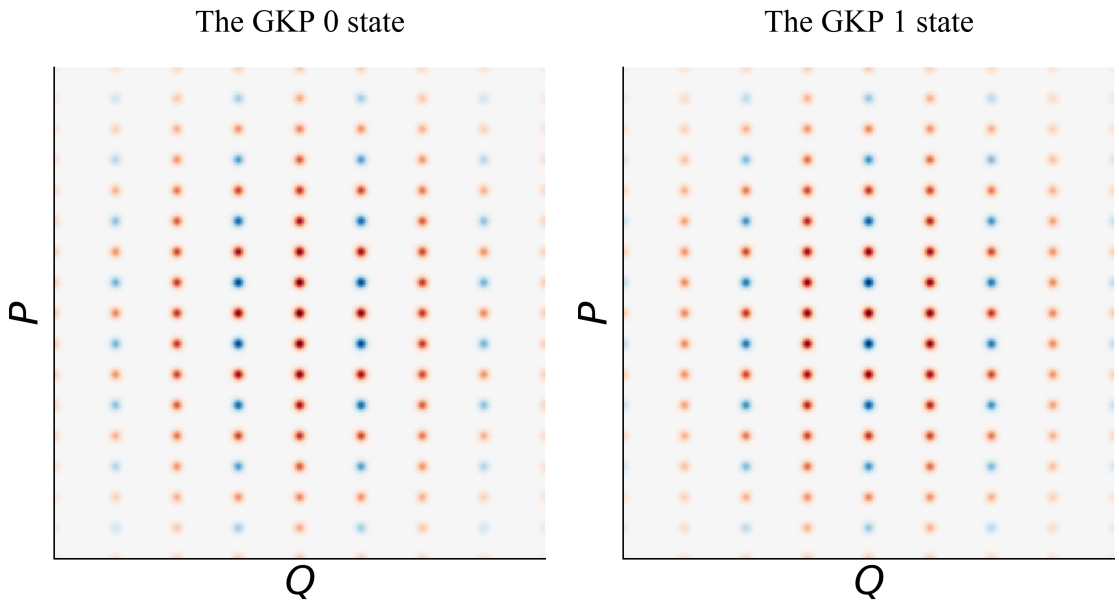


Figure 4.1: Wigner function of two GKP logical states

each δ functions and a bigger Gaussian wave packet(with width$\tilde{\Delta}$) to envelope peak heights. The approximated code writes:

$$|\bar{0}\rangle = N_0 \sum_{s=-\infty}^{\infty} \int_{-\infty}^{\infty} dq e^{-\frac{1}{2\tilde{\Delta}^2}(2s\sqrt{\pi})^2} e^{-\frac{1}{2\Delta^2}(q-2\sqrt{\pi}s)^2} |q\rangle \tag{4.1}$$

where $N_0 \approx (\frac{1}{\pi\Delta^2})^{\frac{1}{4}}$ is the normalization factor. In the following we assume $\Delta = \frac{1}{\tilde{\Delta}}$. The approximated GKP state $|\bar{0}\rangle$ is plotted in Fig. 1(for $\Delta = \frac{1}{\tilde{\Delta}} = 0.2$). Since the p-space wave function is complex, we plotted p wave function in probability density $|\psi|^2$.

The approximated state in p space:

$$\begin{aligned}
|\bar{0}\rangle_{approx} &= (\frac{1}{\pi\Delta^2})^{\frac{1}{4}} \sum_{s=-\infty}^{\infty} \int_{-\infty}^{\infty} dq e^{-\frac{1}{2\tilde{\Delta}^2}(2s\sqrt{\pi})^2} e^{-\frac{1}{2\Delta^2}(q-2\sqrt{\pi}s)^2} |q\rangle \\
&= (\frac{1}{\pi\Delta^2})^{\frac{1}{4}} \sum_{s=-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} dq e^{-2\pi\Delta^2 s^2} e^{-\frac{1}{2\Delta^2}(q-2\sqrt{\pi}s)^2} \frac{1}{\sqrt{2\pi}} e^{ipq} dp |p\rangle \\
&= (\frac{1}{4\pi^3\Delta^2})^{\frac{1}{4}} \sum_{s=-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} dq e^{-2\pi\Delta^2 s^2} e^{-\frac{1}{2\Delta^2}(q-2\sqrt{\pi}s)^2} e^{ipq} dp |p\rangle
\end{aligned}$$

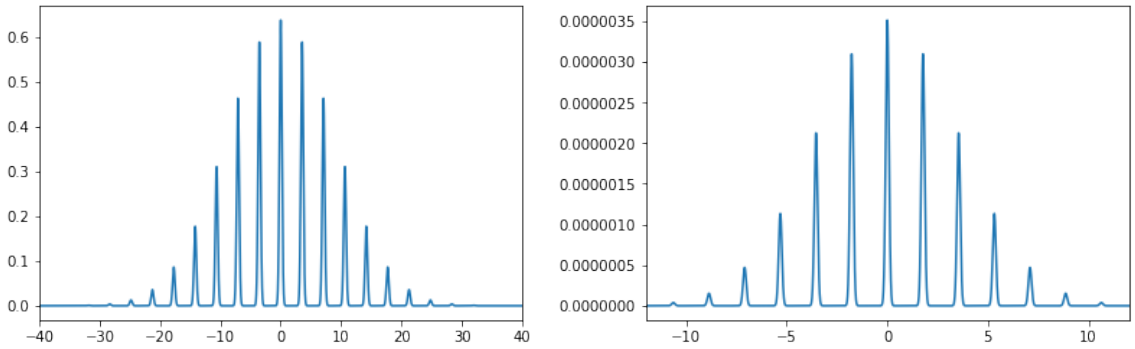


Figure 4.2: Approximated GKP state $|\bar{0}\rangle_{approx}$ in Qutip. Left: q space. Right: probability in p space(without normalization).

$$
\begin{aligned}
&= (\frac{1}{4\pi^3\Delta^2})^{\frac{1}{4}} \sum_{s=-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-2\pi\Delta^2 s^2} \Big[\int_{-\infty}^{\infty} dq e^{-\frac{1}{2\Delta^2}(q-2s\sqrt{\pi}-ip\Delta^2)^2} \Big] \\
&\quad e^{-\frac{p^2\Delta^2}{2}} e^{2ips\sqrt{\pi}} dp \, |p\rangle \\
&= (\frac{1}{4\pi^3\Delta^2})^{\frac{1}{4}} \sum_{s=-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-2\pi\Delta^2 s^2} \Big[\int_{-\infty}^{\infty} dq e^{-\frac{1}{2\Delta^2}(q-ip\Delta^2)^2} \Big] e^{-\frac{p^2\Delta^2}{2}} e^{2ips\sqrt{\pi}} dp \, |p\rangle \\
&= (\frac{1}{4\pi^3\Delta^2})^{\frac{1}{4}} \sum_{s=-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-2\pi\Delta^2 s^2} \Big[\int_{0}^{\infty} dq (e^{-\frac{1}{2\Delta^2}(q-ip\Delta^2)^2} + e^{-\frac{1}{2\Delta^2}(-q-ip\Delta^2)^2}) \Big] \\
&\quad e^{-\frac{p^2\Delta^2}{2}} e^{2ips\sqrt{\pi}} dp \, |p\rangle \\
&= (\frac{1}{4\pi^3\Delta^2})^{\frac{1}{4}} \sum_{s=-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-2\pi\Delta^2 s^2} \Big[\int_{0}^{\infty} dq e^{-\frac{1}{2\Delta^2}(q^2-p^2\Delta^4)} (e^{-qpi} + e^{qpi}) \Big] \\
&\quad e^{-\frac{p^2\Delta^2}{2}} e^{2ips\sqrt{\pi}} dp \, |p\rangle \\
&= 2(\frac{1}{4\pi^3\Delta^2})^{\frac{1}{4}} \sum_{s=-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-2\pi\Delta^2 s^2} \Big[\int_{0}^{\infty} dq cos(qp) e^{-\frac{1}{2\Delta^2}q^2} \Big] e^{2ips\sqrt{\pi}} dp \, |p\rangle \\
&= 2(\frac{1}{4\pi^3\Delta^2})^{\frac{1}{4}} \sum_{s=-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-2\pi\Delta^2 s^2} \Big[\frac{\sqrt{\pi} e^{-\frac{p^2\Delta^2}{2}}}{2\sqrt{\frac{1}{2\Delta^2}}} \Big] e^{2ips\sqrt{\pi}} dp \, |p\rangle \\
&= (\frac{\Delta^2}{\pi})^{\frac{1}{4}} \sum_{s=-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-2\pi\Delta^2 s^2} e^{2ips\sqrt{\pi}} dp e^{-\frac{1}{2}p^2\Delta^2} \, |p\rangle
\end{aligned}
\tag{4.2}
$$

(Use poisson summation rule: $\sum_{t=-\infty}^{\infty} e^{-\pi a(t-b)^2} = (a)^{-\frac{1}{2}} \sum_{s=-\infty}^{\infty} e^{-\frac{\pi s^2}{a}} e^{2\pi isb}$)

$$= \sqrt{2}\Delta(\frac{\Delta^2}{\pi})^{\frac{1}{4}} \sum_{t=-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-\frac{1}{2}p^2\Delta^2} e^{-\frac{1}{2\Delta^2}(p-\sqrt{\pi}t)^2} dp \, |p\rangle$$

$$\approx \sqrt{2}\Delta(\frac{\Delta^2}{\pi})^{\frac{1}{4}} \sum_{t=-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-\frac{1}{2}\pi t^2\Delta^2} e^{-\frac{1}{2\Delta^2}(p-\sqrt{\pi}t)^2} dp \, |p\rangle$$

(because that the peaks are narrow, within each peak, we have $p \approx \sqrt{\pi}t$)

Some remarks:

- In the approximated GKP state, the logical operation $\bar{X} = e^{-i\sqrt{\pi}p}$ no longer has the property of $\bar{X}^2 = I$ unless it's forced back to the code space(through phase estimation or some other means) after each $\bar{X}$ operation.

The mean photon number $\bar{n} = \frac{1}{2}\langle q^2 + p^2\rangle + \frac{1}{2} = \langle q^2\rangle + \frac{1}{2}$ of this approximation of $|\bar{0}\rangle$ can be calculated:

$$\begin{aligned}
\langle 0| \, q^2 \, |0\rangle &= \frac{1}{\sqrt{\pi}\Delta} \sum_{t=-\infty}^{\infty} \sum_{s=-\infty}^{\infty} e^{-2\pi\Delta^2(t^2+s^2)} \int_{-\infty}^{\infty} q^2 e^{-\frac{1}{2\Delta^2}[(q-2t\sqrt{\pi})^2+(q-2s\sqrt{\pi})^2]} dq \, |q\rangle \\
&= \frac{1}{\sqrt{\pi}\Delta} \sum_{t=-\infty}^{\infty} \sum_{s=-\infty}^{\infty} e^{-2\pi\Delta^2(t^2+s^2)} \\
&\quad \int_{-\infty}^{\infty} q^2 e^{-\frac{1}{\Delta^2}[q^2-2(t+s)\sqrt{\pi}q+(t^2+s^2)\pi+2ts\pi+(t^2-2ts+s^2)\pi]} dq \, |q\rangle \\
&= \frac{1}{\sqrt{\pi}\Delta} \sum_{t=-\infty}^{\infty} \sum_{s=-\infty}^{\infty} e^{-2\pi\Delta^2(t^2+s^2)-\frac{1}{\Delta^2}\pi(t-s)^2} \int_{-\infty}^{\infty} q^2 e^{-\frac{1}{\Delta^2}(q-(t+s)\sqrt{\pi})^2} dq \, |q\rangle \\
&= \frac{1}{\sqrt{\pi}\Delta} \sum_{t=-\infty}^{\infty} \sum_{s=-\infty}^{\infty} e^{-2\pi\Delta^2(t^2+s^2)-\frac{1}{\Delta^2}\pi(t-s)^2} \int_{-\infty}^{\infty} (q+(t+s)\sqrt{\pi})^2 e^{-\frac{1}{\Delta^2}q^2} dq \, |q\rangle \\
&= \frac{1}{\sqrt{\pi}\Delta} \sum_{t=-\infty}^{\infty} \sum_{s=-\infty}^{\infty} e^{-2\pi\Delta^2(t^2+s^2)-\frac{1}{\Delta^2}\pi(t-s)^2} \Bigg\{ \int_{-\infty}^{\infty} q^2 e^{-\frac{1}{\Delta^2}q^2} dq \, |q\rangle \\
&\quad + \underbrace{\int_{-\infty}^{\infty} 2(t+s)\sqrt{\pi} q e^{-\frac{1}{\Delta^2}q^2} dq \, |q\rangle}_{0} + \int_{-\infty}^{\infty} (t+s)^2 \pi e^{-\frac{1}{\Delta^2}q^2} dq \, |q\rangle \Bigg\} \\
&= \frac{1}{\sqrt{\pi}\Delta} \sum_{t=-\infty}^{\infty} \sum_{s=-\infty}^{\infty} e^{-2\pi\Delta^2(t^2+s^2)-\frac{1}{\Delta^2}\pi(t-s)^2} \Big(\frac{1}{2}\sqrt{\pi}\Delta^3 + (t+s)^2\sqrt{\pi}\pi\Delta\Big)
\end{aligned}$$

$$\approx \pi \sum_{t=-\infty}^{\infty} \sum_{s=-\infty}^{\infty} e^{-2\pi\Delta^2(t^2+s^2)-\frac{1}{\Delta^2}\pi(t-s)^2}(t+s)^2$$
$$= \frac{1}{2\Delta^2\sqrt{1+\Delta^4}}$$
$$\approx \frac{1}{2\Delta^2}$$

(We used Gaussian integral $\int_{-\infty}^{\infty} e^{-ax^2}dx = \sqrt{\frac{\pi}{a}}$ and $\int_{-\infty}^{\infty} x^2 e^{-ax^2}dx = \frac{1}{2}\sqrt{\frac{\pi}{a^3}}$)

4.3 Prepare a GKP state — the Phase Estimation Protocol

The most efficient way to prepare an approximate GKP code is proposed in [180] using phase estimation.

The phase estimation protocol proposed is a modified version from the original PE [94]. It adopts the "information theory PE" of Freedman, Krista, Hastings [174] to make the implementation feasible(limited number of photons).

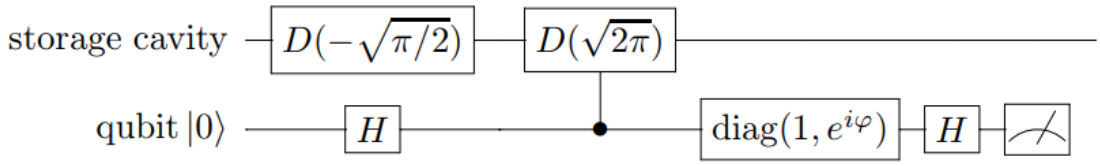


Figure 4.3: IPE

In Fig. 4.3, one round of the protocol is illustrated. Suppose $|\xi_k\rangle$ is an eigenstate of operator U(here $U = S_q = D(\sqrt{2\pi})$ or $S_p = D(\sqrt{2i\pi})$) with eigenvalue $\lambda = e^{2\pi i\theta_k}$ where $\theta_k = \frac{k}{t}$, $0 \geq k < t < 2^m$.

4.3.1 The PE measurement operator of 1 *round*

The measurement operator constructed in the above circuit can be written as:

$$U^{\psi}_{measure} = \frac{1}{2}\begin{pmatrix} D(-\sqrt{\frac{\pi}{2}}) + D(\sqrt{\frac{\pi}{2}})e^{i\psi} & D(-\sqrt{\frac{\pi}{2}}) - D(\sqrt{\frac{\pi}{2}})e^{i\psi} \\ D(-\sqrt{\frac{\pi}{2}}) - D(\sqrt{\frac{\pi}{2}})e^{i\psi} & D(-\sqrt{\frac{\pi}{2}}) + D(\sqrt{\frac{\pi}{2}})e^{i\psi} \end{pmatrix} \tag{4.3}$$

where the parameters ψ and M can be adjusted. In the GKP case, acting on a squeezed vacuum state $|0\rangle \otimes |\text{sq.vac}\rangle$:

$$\begin{aligned} U^{\psi}_{measure} |0 \otimes \text{sq.vac}\rangle &= \frac{1}{2}\Big((|0\rangle + |1\rangle) \otimes D(-\sqrt{\frac{\pi}{2}}) |\text{sq.vac}\rangle + (|0\rangle - |1\rangle)e^{i\psi} \otimes D(\sqrt{\frac{\pi}{2}}) |\text{sq.vac}\rangle \Big) \\ &= \frac{1}{2}\Big\{ |0\rangle \otimes [D(-\sqrt{\frac{\pi}{2}}) + e^{i\psi} D(\sqrt{\frac{\pi}{2}})] |\text{sq.vac}\rangle + \\ &\quad |1\rangle \otimes [D(-\sqrt{\frac{\pi}{2}}) - e^{i\psi} D(\sqrt{\frac{\pi}{2}})] |\text{sq.vac}\rangle \Big\} \end{aligned} \tag{4.4}$$

4.3.2 The PE measurement operator of M rounds

Let's consider the state change after M rounds. For one round, if the ancilla qubit is measured to be Res$= 0$ (or 1), the resulting state becomes:

$$|\tilde{0}\rangle_1 = D(-\sqrt{\frac{\pi}{2}}) |\text{sq.vac}\rangle + (-1)^{\text{Res}} D(\sqrt{\frac{\pi}{2}}) e^{i\psi} |\text{sq.vac}\rangle$$

After repeating the above circuit for M rounds(suppose M is even), we measure a string $x[M]$, we estimate the phase θ with the phase that gives the maximal probability of getting $x[M]$. Suppose for all rounds, $\psi = 0$, the resulting state is only related to the Hamming weight $w_H(x)$, the measurement operator gives:

$$\big[D(-\sqrt{\frac{\pi}{2}}) + D(\sqrt{\frac{\pi}{2}})\big]^{M - w_H(x)} \big[D(-\sqrt{\frac{\pi}{2}}) - D(\sqrt{\frac{\pi}{2}})\big]^{w_H(x)}$$

Clearly if $w_H(x) = 0$, the operator becomes $\big[D(-\sqrt{\frac{\pi}{2}}) + D(\sqrt{\frac{\pi}{2}})\big]^M$ and an approximated GKP state is achieved. This has to be done by post-selection and is discussed previously [187, 77]. However, by post-selection, the successful rate drops exponentially.

In general(assume different ψ each round and no post-selection), M arounds of the above

circuit will give the following state:

$$|\Phi(x[M])\rangle = \frac{1}{\sqrt{N}} \prod_{m=1}^{M} \left[D(-\sqrt{\frac{\pi}{2}}) + e^{i(\psi_m + x_m \pi)} D(\sqrt{\frac{\pi}{2}})\right] |\text{sq.vac}\rangle$$

Where N is the normalization factor. Suppose we want to expand the product. From the the product of M sums, if we choose j of them to be positive shift($D(\sqrt{\frac{\pi}{2}})$), $M-j$ to be negative shift($D(-\sqrt{\frac{\pi}{2}})$), we can produce a peak at $\sqrt{\pi/2}[j-(M-j)] = \sqrt{2\pi}(j-M/2)$ in the q space, and the produced phase is $\prod_{l\in S_j} e^{i(\psi_l + x_l\pi)}$($S_j$ is the set of rounds we choose positive shifts).There are $\binom{M}{m}$ ways to choose such an S_j, by choosing any of such S_j we can produce a peak at $\sqrt{2\pi}(j - M/2)$. Thus, $|\psi(x[M])\rangle$ can be written as:

$$|\Phi(x[M])\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{M} c_j(x[M]) D(\sqrt{2\pi}(j - M/2)) |\text{sq.vac}\rangle \tag{4.5}$$

where $N \approx \sum_{j=0}^{M} |c_j(x[M])|^2$, $c_j(x[M]) = \sum_{\{S_j\}} \prod_{l\in S_j} e^{i(\psi_l + x_l\pi)}$.

4.3.3 What is a good state?

The resulting $|\Phi(x[M])\rangle$(Eq. (4.5)) is not always an approximated GKP state. A natural question to ask is what state is a good state. The hope is that even if we produce a bad state, we can make it a good one by simply shifting it back to the original(That means the bad states are just shifted GKP in p space).

First, notice in the ideal case that $e^{i(\psi_l + x_l\pi)} = 1$ for all l, then $c_j^{\text{ideal}} = \binom{M}{j} \approx e^{-\frac{2(j-\frac{M}{2})^2}{M}}$, which gives us a good approximate GKP state. However, to achieve this, we need post selection. To get a sense how the phase change the resulting state, let's assume there's only one l in Eq. (4.5) such that $e^{i(\psi_l + x_l\pi)} = e^{i\theta_l} \neq 1$. In this case, for the $D(\sqrt{2\pi}(j-M/2)) |\text{sq.vac}\rangle$ peak, among the $\binom{M}{j}$

terms that contribute to the peak height, there are $\binom{M-1}{j-1}$ have the term $e^{i\theta_l}$ and $\binom{M-1}{j}$ have not and be 1(we can verify that $\binom{M-1}{j-1} + \binom{M-1}{j} = \binom{M}{j}$, and $\binom{M-1}{j-1} = \frac{j}{M}c_j^{\text{ideal}}$, $\binom{M-1}{j} = \frac{M-j}{M}c_j^{\text{ideal}}$). Then the probability of measuring peak(or the peak height of) $D(\sqrt{2\pi}(j-M/2))\,|\text{sq.vac}\rangle$ is:

$$\begin{aligned} Pr(D(\sqrt{2\pi}(j-M/2))\,|\text{sq.vac}\rangle) &= \frac{1}{N}(c_j^{ideal})^2(e^{i\theta_l}\frac{j}{M}+\frac{M-j}{M})(e^{-i\theta_l}\frac{j}{M}+\frac{M-j}{M}) \\ &= [(\frac{j}{M})^2+(\frac{M-j}{M})^2+2cos(\theta_l)\frac{j(M-j)}{M^2}]\frac{1}{N}(c_j^{ideal})^2 \end{aligned}$$

If we choose $\theta_l = \pi$, it's easy to see that the central peak(where $M-j=j$) vanishes, which makes the state a bad state. We numerically generate the result for $x[8] = 11111111$ and $\psi[8] = 0\pi\pi\pi\pi\pi\pi\pi$ in Fig. 4.4 to confirm that such state cannot become an approximated GKP state by simply being shifted in p space. If $-\frac{\pi}{2} < \theta_l < \frac{\pi}{2}$, we can make sure that the central peak retains at least half the height. With $x_l\pi = 0$ or π, the safe choice is to choose $\psi_l = -\frac{-pi}{2}$ that guarantees θ_l to be in this range.

Similarly, we can assume that there are only two of the phases $\theta_1, \theta_2 \neq 2\pi$. The height of each peak is then:

$$\begin{aligned} Pr(j-M/2) \propto &\Big(\frac{j(j-1)}{M(M-1)}e^{i(\theta_1+\theta_2)}+\frac{(M-j)(M-j-1)}{M(M-1)}+\frac{(M-j)j}{M(M-1)}(e^{i\theta_1}+e^{i\theta_2})\Big) \\ &\times\Big(\frac{j(j-1)}{M(M-1)}e^{-i(\theta_1+\theta_2)}+\frac{(M-j)(M-j-1)}{M(M-1)}+\frac{(M-j)j}{M(M-1)}(e^{-i\theta_1}+e^{-i\theta_2})\Big) \\ &= \big[\frac{j(j-1)}{M(M-1)}\big]^2+\big[\frac{(M-j)(M-j-1)}{M(M-1)}\big]^2+\frac{j^2(M-j)}{M^2(M-1)}(2cos(\theta_1-\theta_2)+2) \\ &\quad +2\frac{j(j-1)(M-j)(M-j-1)}{M^2(M-1)^2}cos(\theta_1+\theta_2)+ \\ &\quad \frac{j(M-j)^2(M-j-1)}{M^2(M-1)^2}(2cos(\theta_1)+2cos(\theta_2)) \end{aligned}$$

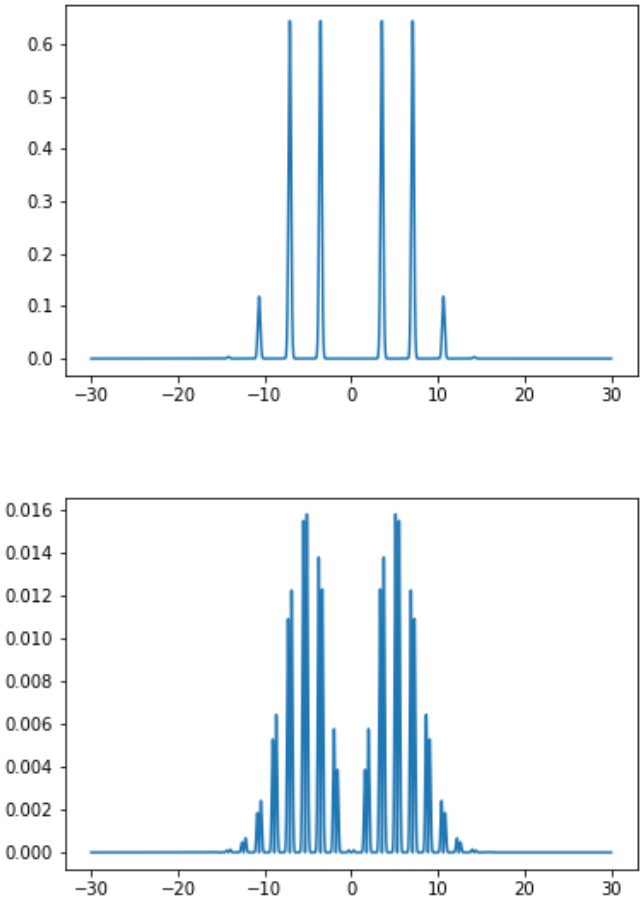


Figure 4.4: Example of a bad state generated by phase estimation($x[8] = 11111111$ and $\psi[8] = 0\pi\pi\pi\pi\pi\pi\pi$). left: q space. right: p space.

The above expression can give us some hint about the strategy of choosing the the next ψ. If we know that previously only $\theta_1 \neq 0 (\text{mod } 2\pi)$, we can choose $\psi_2 = \theta_1$ or $\theta_1 - \pi$ depending on the probability of measuring 0 and measuring 1 this round to maximize the central peak. In Fig. 4.5, we illustrate the end state of $x[8] = 11111111, \psi[8] = \pi\pi\pi\pi\pi\pi 0.5\pi 0.5\pi$ and $x[8] = 11111110, \psi[8] = \pi\pi\pi\pi\pi\pi 0.5\pi 0.5\pi$.

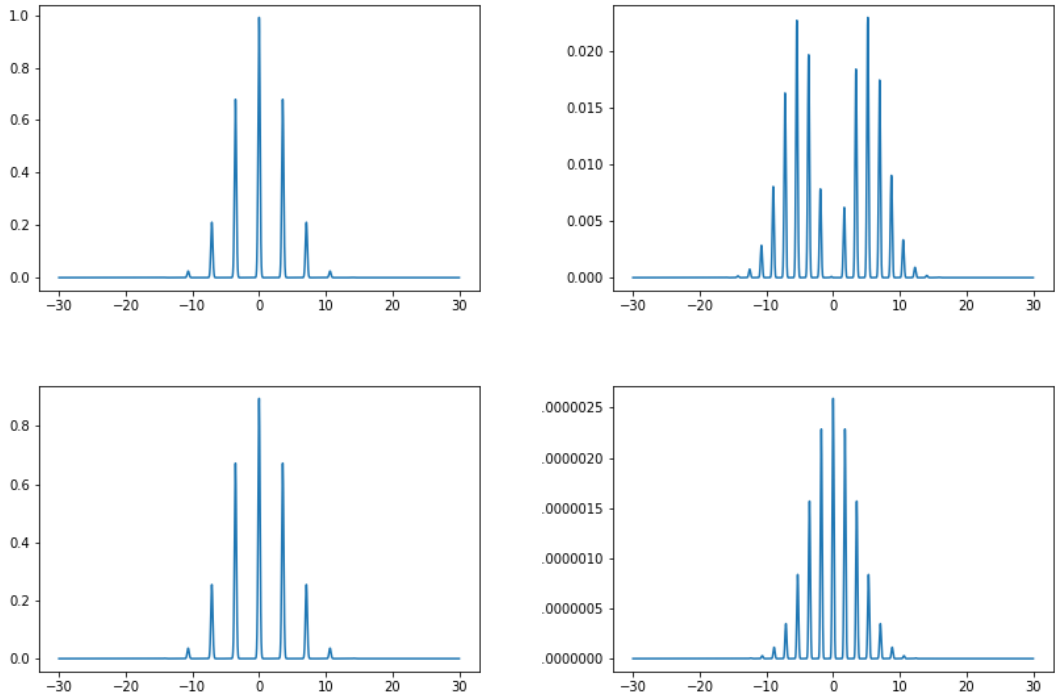


Figure 4.5: Above: $x[8] = 11111111, \psi[8] = \pi\pi\pi\pi\pi\pi 0.5\pi 0.5\pi$ in q and p space. Bottom: $x[8] = 11111110, \psi[8] = \pi\pi\pi\pi\pi\pi 0.5\pi 0.5\pi$ in q and p space.

4.3.4 The probability of measuring 0 *or* 1 *at each round*

In principle, we can decompose $|\text{sq.vac}\rangle$ into $\sum_k C_k \lambda_k$, where λ_k are eigenstates of the displacement operators:

$$
\begin{aligned}
U^{\theta}_{measure} \, |0 \otimes \text{sq.vac}\rangle = \frac{1}{2} \Big\{ & |0\rangle \otimes [D(-\sqrt{\frac{\pi}{2}}) + e^{i\psi} D(\sqrt{\frac{\pi}{2}})]) \, |\text{sq.vac}\rangle \\
& |1\rangle \otimes [D(-\sqrt{\frac{\pi}{2}}) - e^{i\psi} D(\sqrt{\frac{\pi}{2}})]) \, |\text{sq.vac}\rangle \Big\} \\
= \frac{1}{2} \Big[& \sum_k (e^{-\frac{1}{2} i\theta_k} C_k + e^{i(\psi + \frac{1}{2}\theta_k)} C_k) \, |0\rangle \otimes \lambda_k + \\
& \sum_k (e^{-\frac{1}{2}\theta_k} C_k - e^{i(\psi + \frac{1}{2}\theta_k)} C_k) \, |1\rangle \otimes C_k \lambda_k \Big]
\end{aligned}
$$

The probability of measuring 0 is:

$$\begin{aligned} P(Res=0) &= \sum_k \frac{1}{2}(1+cos(\theta_k+\psi))|C_k|^2 \\ &\approx \frac{1}{2}(1+cos(\theta+\psi)) = cos^2(\frac{\theta}{2}+\frac{\psi}{2}) \end{aligned}$$

Explicit calculation of the probability

Alternatively, the probability of measuring 0 at the end of the first round can be written as:

$$\begin{aligned} P(Res=0) &= \langle 0\otimes \text{sq.vac}|(U^{\psi}_{measure})^{\dagger}|0\otimes I\rangle \langle 0\otimes I|U^{\psi}_{measure}|0\otimes \text{sq.vac}\rangle \\ &= \frac{1}{4}\langle \text{sq.vac}[D(-\sqrt{\frac{\pi}{2}})+e^{-i\psi}D(\sqrt{\frac{\pi}{2}})]\Big|[D(-\sqrt{\frac{\pi}{2}})+e^{i\psi}D(\sqrt{\frac{\pi}{2}})]\text{sq.vac}\rangle \\ &\quad (\text{because } \langle 0|1\rangle = \langle 1|0\rangle = 0, \text{the } |1\rangle \text{ part is eliminated.}) \\ &= \frac{1}{4\sqrt{\pi}\Delta}\int_{-\infty}^{\infty}\Bigg\{e^{-\frac{1}{2\Delta^2}2(q-\sqrt{\pi})^2}+ \\ &\quad e^{-\frac{1}{2\Delta^2}2(q+\sqrt{\pi})^2}+(e^{i\psi}+e^{-i\psi})e^{-\frac{1}{2\Delta^2}[(q-\sqrt{\pi})^2+(q+\sqrt{\pi})^2]}\Bigg\}dq \\ &= \frac{1}{4\sqrt{\pi}\Delta}\left\{\sqrt{\pi}\Delta+\sqrt{\pi}\Delta+2cos(\psi)\int_{-\infty}^{\infty}e^{-\frac{1}{\Delta^2}(q^2+\pi)}dq\right\} \\ &= \frac{1}{2}+\frac{cos(\psi)e^{-\frac{\pi}{\Delta^2}}}{2} \end{aligned} \tag{4.6}$$

The probability of measuring 1 can be obtained similarly. With knowing that the Gaussian width Δ is small, the probability is approximately 0.5(For example, if $\Delta = 0.2$, $cos(\psi)e^{-\pi/\Delta^2} < 10^{-35}$). Let's also calculate the probability of measuring 0 again after the second round.

The input state to the second round is(assuming measured 0 at the end of the first round):

$$\begin{aligned} |\Phi_{\text{input}}\rangle_{2nd} &= \frac{|0\otimes I\rangle\langle 0\otimes I|U^{\psi_1}_{measure}|0\otimes \text{sq.vac}\rangle}{\sqrt{P(Res=0)}} \\ &= \frac{1}{2}\sqrt{\frac{2}{1+cos(\psi_1)e^{-\frac{\pi}{\Delta^2}}}}\,|0\otimes[D(-\sqrt{\frac{\pi}{2}})+e^{i\psi_1}D(\sqrt{\frac{\pi}{2}})]\text{sq.vac}\rangle \end{aligned}$$

Then the probability is given by:

$$
\begin{aligned}
Pr_{2nd}(Res=0) &= \langle \Phi_{\text{input}}|0\otimes I\rangle \langle 0\otimes I|\Phi_{\text{input}}\rangle_{2nd} \\
&= \frac{1}{8+8cos(\psi_1)e^{-\frac{\pi}{\Delta^2}}} \\
&\langle \text{sq.vac}[D(-\sqrt{\frac{\pi}{2}})+e^{-i\psi_1}D(\sqrt{\frac{\pi}{2}})][D(-\sqrt{\frac{\pi}{2}})+e^{-i\psi_2}D(\sqrt{\frac{\pi}{2}})]| \\
&|[D(-\sqrt{\frac{\pi}{2}})+e^{i\psi_2}D(\sqrt{\frac{\pi}{2}})][D(-\sqrt{\frac{\pi}{2}})+e^{i\psi_1}D(\sqrt{\frac{\pi}{2}})]\text{sq.vac}\rangle \\
&= \frac{1}{8+8cos(\psi_1)e^{-\frac{\pi}{\Delta^2}}} \\
&\langle \text{sq.vac}[D(-\sqrt{2\pi})+(e^{-i\psi_1}+e^{-i\psi_2})I+e^{-i(\psi_1+\psi_2)}D(\sqrt{2\pi})]| \\
&|[D(-\sqrt{2\pi})+(e^{i\psi_1}+e^{i\psi_2})I+e^{i(\psi_1+\psi_2)}D(\sqrt{2\pi})]\text{sq.vac}\rangle \\
&= \frac{1}{8+8cos(\psi_1)e^{-\frac{\pi}{\Delta^2}}}\int_{-\infty}^{\infty}dq \\
&\Big\{e^{-\frac{1}{2\Delta^2}2(q+2\sqrt{\pi})^2}+(e^{i\psi_1}+e^{i\psi_2})e^{-\frac{1}{2\Delta^2}[(q+2\sqrt{\pi})^2+q^2]}+ \\
&e^{i(\psi_1+\psi_2)}e^{-\frac{1}{2\Delta^2}[(q+2\sqrt{\pi})^2+(q-2\sqrt{\pi})^2]} \\
&+(e^{-i\psi_1}+e^{-i\psi_2})e^{-\frac{1}{2\Delta^2}[(q+2\sqrt{\pi})^2+q^2]}+ \\
&(e^{-i\psi_1}+e^{-i\psi_2})(e^{i\psi_1}+e^{i\psi_2})e^{-\frac{1}{2\Delta^2}2q^2} \\
&+(e^{-i\psi_1}+e^{-i\psi_2})e^{i(\psi_1+\psi_2)}e^{-\frac{1}{2\Delta^2}[(q-2\sqrt{\pi})^2+q^2]} \\
&+e^{-i(\psi_1+\psi_2)}e^{-\frac{1}{2\Delta^2}[(q+2\sqrt{\pi})^2+(q-2\sqrt{\pi})^2]}+ \\
&e^{-i(\psi_1+\psi_2)}(e^{i\psi_1}+e^{i\psi_2})e^{-\frac{1}{2\Delta^2}[(q-2\sqrt{\pi})^2+q^2]} \\
&+e^{-i(\psi_1+\psi_2)}e^{i(\psi_1+\psi_2)}e^{-\frac{1}{2\Delta^2}2(q-2\sqrt{\pi})^2}\Big\} \\
&= \frac{1}{4+4cos(\psi_1)e^{-\frac{\pi}{\Delta^2}}}[2+cos(\psi_1-\psi_2)+2e^{\frac{-\pi}{\Delta^2}}(cos(\psi_1)+cos(\psi_2))+ \\
&e^{\frac{-4\pi}{\Delta^2}}cos(\psi_1+\psi_2)] \qquad (4.7)
\end{aligned}
$$

The above probability is approximately $\frac{2+cos(\psi_1-\psi_2)}{4}$.

Now we turn to the probability of measuring $x_M (x_M = 0 \text{ or } 1)$ after M rounds. Denote the probability of measuring x_k at round k as Pr_{x_k}. Then what we are interested is Pr_{x_M}. We assume we already performed $M-1$ rounds of PE and recorded the measurement results $x_1, ..., x_{M-1}$ and angle parameters we chose (denote as $\psi_1, ...\psi_{M-1}$). Then the input state to the Mth round is:

$$|\Phi_{input}\rangle_M = \frac{1}{2^{M-1}\sqrt{Pr_{x_1}...Pr_{x_{M-1}}}} \sum_{j=0}^{M-1} c_j(x[M-1]) D(\sqrt{2\pi}(j-(M-1)/2)) |0\rangle \otimes |\text{sq.vac}\rangle$$

where c_j is consistent with Eq. (4.5).

Then the probability of measuring x_M is:

$$\begin{aligned}
Pr_{x_M} &= \langle \Phi_{input}| (U^{\psi_M}_{measure})^\dagger |x_M \otimes I\rangle \langle x_M \otimes I| U^{\psi_M}_{measure} |\Phi_{input}\rangle_M \\
&= \Big| \frac{1}{2^M \sqrt{Pr_{x_1}...Pr_{x_{M-1}}}} \sum_{j=0}^{M} c_j(x[M]) D(\sqrt{2\pi}(j-(M)/2)) |\text{sq.vac}\rangle \Big|^2 \\
&\quad \text{(From the calculation of the first 2 rounds, we notice that the transfer probability from one peak to another is almost 0}(< 10^{-35}\text{), then we can make the following approximation.)} \\
&\approx \frac{1}{2^{2M} Pr_{x_1}...Pr_{x_{M-1}}} \sum_{j=0}^{M} \Big| c_j(x[M]) D(\sqrt{2\pi}(j-(M)/2)) |\text{sq.vac}\rangle \Big|^2 \\
&= \frac{1}{2^{2M} Pr_{x_1}...Pr_{x_{M-1}}} \sum_{j=0}^{M} \Big| c_j(x[M]) \Big|^2 \\
&\quad \underbrace{\langle \text{sq.vac} D^\dagger(\sqrt{2\pi}(j-(M)/2)) | D(\sqrt{2\pi}(j-(M)/2)) \text{sq.vac}\rangle}_{1} \\
&= \frac{1}{2^{2M} Pr_{x_1}...Pr_{x_{M-1}}} \sum_{j=0}^{M} \Big| \sum_{\{S_j\}} \prod_{l \in S_j} e^{i(\psi_l + x_l \pi)} \Big|^2 \\
&= \frac{1}{2^{2M} Pr_{x_1}...Pr_{x_{M-1}}} \sum_{j=0}^{M} \Big[\Big(\sum_{\{S_j^m\}} \prod_{l \in S_j^m} e^{-i(\psi_l + x_l \pi)} \Big) \Big(\sum_{\{S_j^n\}} \prod_{k \in S_j^n} e^{i(\psi_k + x_k \pi)} \Big) \Big]
\end{aligned}$$

The above expression is recursive, we could substitute $Pr_{x_{M-1}}$ into it,

$$Pr_{x_M} = \frac{\sum_{j=0}^{M} \Big[\Big(\sum_{\{S_j^m\}} \prod_{l \in S_j^m} e^{-i(\psi_l + x_l \pi)} \Big) \Big(\sum_{\{S_j^n\}} \prod_{k \in S_j^n} e^{i(\psi_k + x_k \pi)} \Big) \Big]}{4 \sum_{j=0}^{M-1} \Big[\Big(\sum_{\{S_j^m\}} \prod_{l \in S_j^m} e^{-i(\psi_l + x_l \pi)} \Big) \Big(\sum_{\{S_j^n\}} \prod_{k \in S_j^n} e^{i(\psi_k + x_k \pi)} \Big) \Big]}$$

Lastly, we address that the two ways of calculting the probablity agree in practice though having different expressions.

4.3.5 PE strategies

There are roughly two choices for the rotation angles in each round: non-adaptive(choose $\psi = 0, \frac{\pi}{2}$) and adaptive(choose next ψ based on the previous measurement). Here we propose a adaptive PE strategy and compare to Barbara's adaptive PE.

Non-adaptive PE ψ at each round is chosen to be $0, 0, 0, 0, \frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2}$.

Adaptive PE ψ at each round is chosen to maximize the phase resolution,

$$\psi_m = \arg max_{\psi} \sum_{x_m = 0,1} \Big| \int d\theta e^{i\theta} P_{\psi}(x[m]|\theta) \Big|$$

where $P_{\psi}(x[m]|\theta)$ is the probability of measuring $x[m]$ given ψ and θ. Assuming each round is independent, $P_{\psi}(x[m]|\theta)$ can be decomposed into the product of the probability of each round:

$$P_{\psi}(x[m]|\theta) = \prod_{i=1}^{m} P_{\psi_i}(x_i|\theta) = \prod_{i=1}^{m} cos^2(\frac{\theta + \psi_i}{2} + x_i \frac{\pi}{2})$$

An the end of the protocol, the recovery strategy is to apply a shift of $e^{\frac{i\tilde{\theta}}{2\sqrt{\pi}}}$, where $\tilde{\theta}$ is given by,

$$\tilde{\theta} = \arg \int_{-\pi}^{\pi} d\theta e^{i\theta} P_{\psi}(x[m]|\theta)$$

4.4 Goodness of Approximate GKP States

As was explained in [70, 179], preparing perfect GKP states would require an infinite amount of squeezing. In a realistic setting, one can only prepare approximate GKP states with finite squeezing. Perfect GKP states, which are $+1$ eigenstates of the mutually commuting operators $S_p = e^{-2i\sqrt{\pi}p}$ and $S_q = e^{2i\sqrt{\pi}q}$, can correct shift errors of size at most $\frac{\sqrt{\pi}}{2}$. Note that for the displacement operator $D(\alpha) = e^{\alpha a^\dagger - \alpha^* a}$, we can write $S_p = D(\sqrt{2\pi})$ and $S_q = D(i\sqrt{2\pi})$. The goal of this paper will be to fault-tolerantly prepare approximate GKP states which can correct small shift errors correctable by perfect GKP states with high probability (in Section 4.5 we will specify what we mean by correctable shift errors). Therefore, it is important to have a metric which allows us to compute the "goodness" of an approximate GKP state.

Recall that for perfect GKP states, the logical $|\overline{0}\rangle$ and $|\overline{1}\rangle$ states are given by

$$\begin{aligned} |\overline{0}\rangle &= \sum_{k=-\infty}^{\infty} S_p^k \, |q = 0\rangle \\ |\overline{1}\rangle &= \sum_{k=-\infty}^{\infty} S_p^k \, |q = \sqrt{\pi}\rangle \, , \end{aligned} \tag{4.8}$$

up to normalization. In practice, approximate GKP states analogous to those in Eq. (4.8) can be prepared by first preparing finitely squeezed states in q space and approximately projecting these states onto the $S_p = 1$ eigenspace. For instance, the $|q = 0\rangle$ state can be written as a squeezed vacuum state $|\text{sq}\rangle$ with squeezing parameter Δ which is given by

$$|\text{sq}\rangle = \int \frac{dq}{(\pi\Delta^2)^{1/4}} e^{-q^2/(2\Delta^2)} \, |q\rangle \, . \tag{4.9}$$

One can then apply a sum of displacements with a Gaussian filter to obtain

$$|\overline{0}\rangle_{\text{approx}} = C \sum_{k=-\infty}^{\infty} e^{-2\pi\tilde{\Delta}^2 k^2} D(k\sqrt{2\pi}) \, |\text{sq}\rangle \, , \tag{4.10}$$

where C is a normalization coefficient. As was shown in [179], it is natural to have $\tilde{\Delta} = \Delta$. In Sec-

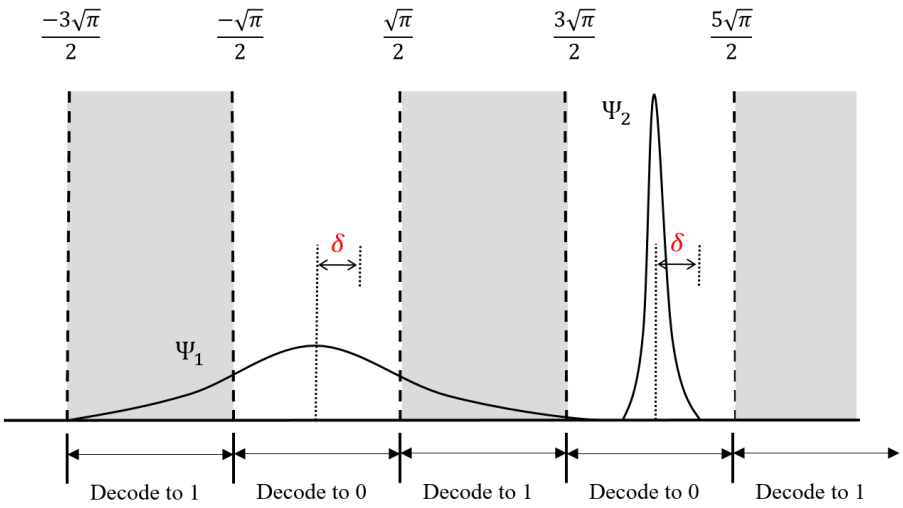


Figure 4.6: Peaks centered at even integer multiples of $\sqrt{\pi}$ in q space. The peak on the left contains large tails that extend into the region where a shift error is decoded to the logical $|\overline{1}\rangle$ state. The peak on the right is much narrower. Consequently for some interval δ, the peak on the right will correct shift errors of size $\frac{\sqrt{\pi}}{2} - \delta$ with higher probability than the peak on the left.

tion 4.6, we will present a fault-tolerant version of phase estimation for approximately projecting the state in Eq. (4.9) onto the $+1$ eigenspace of S_p (see [179] which provides the first description for using phase estimation to prepare approximate GKP states).

Naturally because of the finite width of the peaks of approximate GKP states, it will not be possible to correct a shift error in p or q of magnitude at most $\frac{\sqrt{\pi}}{2}$ with certainty. For example, suppose we have an approximate $|\overline{0}\rangle$ GKP state with a peak at $q = 0$ subject to a shift error e^{-ivp} with $|v| \leq \frac{\sqrt{\pi}}{2}$. The finite width of the Gaussian peaks will have a non-zero overlap in the region $\frac{\sqrt{\pi}}{2} < q < \frac{3\sqrt{\pi}}{2}$ and $\frac{-3\sqrt{\pi}}{2} < q < \frac{-\sqrt{\pi}}{2}$. Thus with non-zero probability the state can be decoded to $|\overline{1}\rangle$ instead of $|\overline{0}\rangle$ (see Fig. 4.6 for an illustration).

In general, if an approximate GKP state is afflicted by a correctable shift error, we would like the probability of decoding to the incorrect logical state to be as small as possible. A smaller overlap of the approximate GKP state in regions in q and p space that lead to decoding the state to the wrong logical state will lead to a higher probability of correcting a correctable shift error by a perfect GKP state. These remarks motivate the following definition

Definition 10 *Let $|\tilde{0}\rangle$ be an approximate logical $|\overline{0}\rangle$ GKP state. We say that $|\tilde{0}\rangle$ is $(\frac{\sqrt{\pi}}{2} - \delta, \epsilon)_q$-GKP*

correctable if and only if for a given tuple (δ, ϵ) *with* $\delta \leq \frac{\sqrt{\pi}}{2}$*, and* $0 \leq \epsilon \leq 1$*, we have*

$$\sum_{k=-\infty}^{\infty} \int_{2\sqrt{\pi}k-\delta}^{2\sqrt{\pi}k+\delta} |\langle q|\tilde{0}\rangle|^2 dq > 1-\epsilon. \tag{4.11}$$

We say that $|\tilde{0}\rangle$ *is* $(\frac{\sqrt{\pi}}{2}-\delta, \epsilon)_p$*-GKP correctable if and only if for a given tuple* (δ, ϵ) *with* $\delta \leq \frac{\sqrt{\pi}}{2}$*, and* $0 \leq \epsilon \leq 1$*, we have*

$$\sum_{k=-\infty}^{\infty} \int_{\sqrt{\pi}k-\delta}^{\sqrt{\pi}k+\delta} |\langle p|\tilde{0}\rangle|^2 dp > 1-\epsilon. \tag{4.12}$$

Note that the bounds in Eqs. (4.11) and (4.12) are different since GKP states have peaks defined on a rectangular lattice. Similarly, for an approximate logical $|\overline{+}\rangle$ state, we have the following definition

Definition 11 *Let* $|\tilde{\overline{+}}\rangle$ *be an approximate logical* $|\overline{+}\rangle$ *GKP state. We say that* $|\tilde{\overline{+}}\rangle$ *is* $(\frac{\sqrt{\pi}}{2}-\delta, \epsilon)_p$*-GKP correctable if and only if for a given tuple* (δ, ϵ) *with* $\delta \leq \frac{\sqrt{\pi}}{2}$*, and* $0 \leq \epsilon \leq 1$*, we have*

$$\sum_{k=-\infty}^{\infty} \int_{2\sqrt{\pi}k-\delta}^{2\sqrt{\pi}k+\delta} |\langle p|\tilde{\overline{+}}\rangle|^2 dp > 1-\epsilon. \tag{4.13}$$

We say that $|\tilde{\overline{+}}\rangle$ *is* $(\frac{\sqrt{\pi}}{2}-\delta, \epsilon)_q$*-GKP correctable if and only if for a given tuple* (δ, ϵ) *with* $\delta \leq \frac{\sqrt{\pi}}{2}$*, and* $0 \leq \epsilon \leq 1$*, we have*

$$\sum_{k=-\infty}^{\infty} \int_{\sqrt{\pi}k-\delta}^{\sqrt{\pi}k+\delta} |\langle q|\tilde{\overline{+}}\rangle|^2 dq > 1-\epsilon. \tag{4.14}$$

As an example, suppose we have two approximate GKP states $|\tilde{0}\rangle_1$ and $|\tilde{0}\rangle_2$ which are $(\frac{\sqrt{\pi}}{2}-\delta, \epsilon_1)_q$ and $(\frac{\sqrt{\pi}}{2}-\delta, \epsilon_2)_q$ correctable for a shift of size $e^{i(\frac{\sqrt{\pi}}{2}-\delta)p}$ (which is correctable by a perfect $|\overline{0}\rangle$ GKP state). If $\epsilon_1 < \epsilon_2$, then $|\tilde{0}\rangle_1$ will correct a shift of size $\frac{\sqrt{\pi}}{2}-\delta$ with greater probability

than $|\tilde{0}\rangle_2$. This is due to the fact that $|\tilde{0}\rangle_1$ has a smaller overlap in regions which result in decoding the logical $|\overline{0}\rangle$ state to the logical $|\overline{1}\rangle$. In this sense we say that $|\tilde{0}\rangle_1$ is better than $|\tilde{0}\rangle_2$ at correcting shift errors in q space.

4.5 Fault-tolerant Definitions

Given a protocol for preparing an approximate GKP state, errors that occur during the protocol can potentially accumulate resulting in a large shift error in addition to deforming the output state. This can then result in an output state which significantly differs from a good approximate GKP state. By good, we mean a state that for a desired value of δ, the state is $(\frac{\sqrt{\pi}}{2} - \delta, \epsilon_1)_q$ and $(\frac{\sqrt{\pi}}{2} - \delta, \epsilon_2)_q$ correctable with $\epsilon_1, \epsilon_2 \ll 1$.

The desired values for δ depend on the particular fault-tolerant error correction protocol used to correct shift errors on encoded data qubits (see for instance [32, 6, 21, 22, 23]). For example, one can use a version of the error correction scheme (which reduces to Steane error correction for qubit CSS stabilizer codes) as proposed by Glancy and Knill in [60]. In this scheme, logical $|\overline{0}\rangle$ and $|\overline{+}\rangle$ ancillas are prepared and interact with the encoded data qubit via a CNOT gate to correct the shift errors afflicting the data qubit. It is shown that the largest shift error that can be corrected is $\frac{\sqrt{\pi}}{6}$. The threshold of $\frac{\sqrt{\pi}}{6}$ arises from how shift errors that are initially afflicting the encoded data qubit and ancilla states propagate through the CNOT gates and combine prior to the measurement. In practice, if additional shift errors occur during the the error correction scheme (say after applying the CNOT gates), the largest correctable shift error could potentially be smaller than $\frac{\sqrt{\pi}}{6}$ [1]. In what follows, we will assume that after preparing the desired ancilla states using some state preparation protocol, the ancillas will be used in the error correction scheme of [60] (assumed to be fault-free) to correct shift errors on encoded data qubits. We also point out that due to the propagation of shift errors in the error correction scheme of [60], it is important to prepare approximate $|\tilde{0}\rangle$ and $|\tilde{+}\rangle$ states that have small shift errors in *both* p and q.

1. Using the optimizations considered in [32], using Knill error correction could potentially increase the threshold of $\frac{\sqrt{\pi}}{6}$ to a larger value.

If a small number of errors that occur during the preparation of an approximate GKP state result in large shift errors (or linear combinations of large shift errors) on the output state, then clearly the protocol used would not be practical. Thus it is important that a given state preparation protocol be fault-tolerant. In what follows we will define what we mean by fault-tolerant. We start with the following two definitions:

Definition 12 *A shift error is said to be correctable if the magnitude of the shift is less than or equal to* $\frac{\sqrt{\pi}}{6}$*. Otherwise, we will say that the shift error is uncorrectable.*

Definition 13 *Suppose we have a protocol for preparing an approximate GKP state. We will say that the output state is an ideal approximate GKP state if no faults occur during the protocol.*

Thus by Definition 13, any approximate GKP state obtained from a state preparation protocol will be called an ideal approximate GKP state if the protocol is implemented fault free, even if the output state is $(\frac{\sqrt{\pi}}{2} - \delta, \epsilon)_{p,q}$ correctable for some desired δ with large ϵ (so that the probability of correcting a shift $\frac{\sqrt{\pi}}{2} - \delta$ is small).

Note that the notion of correctable in Definition 12 assumes that only the data and ancilla qubits used in the error correction scheme of [60] have shift errors. If other operations such as the CNOT gates introduce additional errors, the correctable threshold would be smaller than $\sqrt{\pi}/6$.

With the above definitions we are now ready to define what it means for a state preparation protocol of an approximate GKP state to be fault-tolerant. We point out that in Section 4.6 we consider a fault-tolerant state preparation protocol based on phase estimation. Hence the definitions given below are specific to the case where an approximate GKP state is obtained by coupling a qubit to an oscillator.

Definition 14 $(m, \tilde{\delta})$*-Fault-tolerant state preparation of an approximate GKP state:*

Suppose we have a protocol for preparing an approximate GKP state which is obtained by coupling qubits to a harmonic oscillator. Suppose also that at most m *faults occur during the protocol on the qubit Hilbert space and in addition, a correctable shift error in either* p *or* q *of size*

at most $\tilde{\delta}$ occurs on the oscillator Hilbert space. We will say that the protocol is an $(m, \tilde{\delta})$-Fault-tolerant state preparation of an approximate GKP state protocol if the output state differs from an ideal approximate GKP state by a correctable shift error.

A few clarifications are necessary. Firstly, a fault on the qubit Hilbert space corresponds to a location where an error can occur (see for instance [7]). By location we are referring to a particular time step where either a gate is implemented, a qubit is prepared, a qubit is measured or a qubit is idling. On the oscillator Hilbert space, if an error occurs, we can always expand that error into shift errors (see for instance Eq. (7.12) in [4] and also [179]). By performing the error correction scheme of [60], measuring the q and p quadratures to perform error correction will always project the state onto a state with a single shift error in q and p. Lastly, we point out that $\tilde{\delta}$ in Definition 14 relates to the largest allowed size of the shift error which occurs during the protocol. This should not be confused with δ in Definitions 10 and 11 which relates to the size of a shift error that can be corrected by an approximate GKP state[2].

Intuitively, a fault-tolerant protocol should have the property that if both the number of qubit errors and the size of shift errors are small, the resulting shift error on the output state should be correctable. Depending on the desired value for $\tilde{\delta}$, a particular fault-tolerant error correction protocol might be less tolerant to the size of the shift errors that occurred during the preparation of the approximate GKP state (in practice a different error correction scheme than Steane error correction could be used).

Suppose a state preparation protocol satisfies Definition 14. If during the preparation of the state, m faults occur on the qubit Hilbert space in addition to a shift error of size at most $\tilde{\delta}$ on the oscillator Hilbert space, the remaining shift error on the output state will be corrected in a perfect version of the error correction protocol in [60]. As we will see in Section 4.6, due to measurement errors, the phase estimation protocol presented in [179] needs to be modified in order to be a $(1, \tilde{\delta})$ fault-tolerant protocol. There are additional fault locations (apart from measurement errors) which can result in large shift errors that need to be treated with care.

2. The probability $1 - \epsilon$ of correcting a shift of size $\frac{\sqrt{\pi}}{2} - \delta$ depends on the location and width of the peaks.

Lastly we describe how we will evaluate the error correction properties of an approximate GKP state obtained from a noisy state preparation protocol. It is important to compare the goodness of a GKP state prepared from a noisy circuit to that of an ideal circuit. If the output state of a noisy state preparation protocol is correctable, the shift error will be removed when performing error correction. Therefore we will compare the $(\frac{\sqrt{\pi}}{2} - \delta, \epsilon)_{p,q}$ correctable properties of output states after performing an optimal shift back correction (as long as the shift error is correctable) to the output state. If the shift error is not correctable, the protocol will be deemed too noisy.

The optimal shift back correction is found as follows. In q space, the optimal shift back c^q_{max} is computed as

$$c^q_{\text{max}} =_c \sum_{k=-\infty}^{\infty} \int_{2\sqrt{\pi}k+c-\delta}^{2\sqrt{\pi}k+c+\delta} |\langle q|\tilde{0}\rangle|^2 dq. \tag{4.15}$$

Similarly, in p space we have

$$c^p_{\text{max}} =_c \sum_{k=-\infty}^{\infty} \int_{\sqrt{\pi}k+c-\delta}^{\sqrt{\pi}k+c+\delta} |\langle p|\tilde{0}\rangle|^2 dp. \tag{4.16}$$

For protocols preparing $|\tilde{+}\rangle$, the metric can be defined analogously, but with the integral bounds for p and q switched.

4.6 Fault-tolerant Phase Estimation Protocol

In Section 4.6.1 we will give a brief review of the phase estimation protocol presented in [179]. In Section 4.6.2, we will show how the protocol can be modified so that it becomes a $(1, \tilde{\delta})$-Fault-tolerant state preparation of an approximate GKP state and we will provide the value of $\tilde{\delta}$.

4.6.1 Brief review of the phase estimation protocol for preparing approximate GKP states

We summarize some of properties of the phase estimation protocol introduced in Section 4.3 that are important to our fault-tolerant protocol. The phase estimation circuit for preparing an approximate $|\tilde{0}\rangle$ GKP state is given in Fig. 4.7. The H gate is the Hadamard gate and $\Lambda(e^{i\gamma}) = \text{diag}(1, e^{i\gamma})$. After applying several rounds of the circuit in Fig. 4.7, the input squeezed vacuum state (given in Eq. (4.9)) is projected onto an approximate eigenstate of S_p with some random eigenvalue[3] $e^{i\theta}$. Additionally, an estimated value for the phase θ is obtained. After computing the phase, the state can be shifted back to an approximate $+1$ eigenstate of S_p.

There are a variety of ways to choose the phases γ at each round in the gate $\Lambda(e^{i\gamma})$. In a non-adaptive phase estimation protocol, half of the values for γ can be chosen to be 0 and the other half can be chosen to be $\pi/2$. In an adaptive phase estimation protocol, if the phase to be estimated is θ and assuming no prior knowledge of θ, during the first round the phase can be chosen as $\gamma_1 = 0$. For later rounds, it was shown in [179] that the next phases can be chosen as

$$\gamma_m =_\gamma \sum_{x_m=0,1} \Big| \int d\theta e^{i\theta} P_\gamma(x[m]|\theta) \Big|. \tag{4.17}$$

In Eq. (4.17), the probability $P_\gamma(x[m]|\theta)$ is the probability of obtaining measurement outcomes $x_1, \cdots, x_m$ when the produced output eigenstate $|\psi_\theta\rangle$ has eigenvalue $e^{i\theta}$. Since the measurement results for each round are independent, the estimated probability is given by

$$P_{\gamma=\gamma_m}(x[m]|\theta) = \prod_{i=1}^{m} \cos^2\Big(\frac{\theta+\gamma_i}{2} + x_i\frac{\pi}{2}\Big). \tag{4.18}$$

By analytically computing the evolution of the state, the exact probability of obtaining the outcomes $x[M]$ is derived in Section B.2.5.

3. The phase θ that is obtained depends on the measurement outcome of the ancilla qubit in each round.

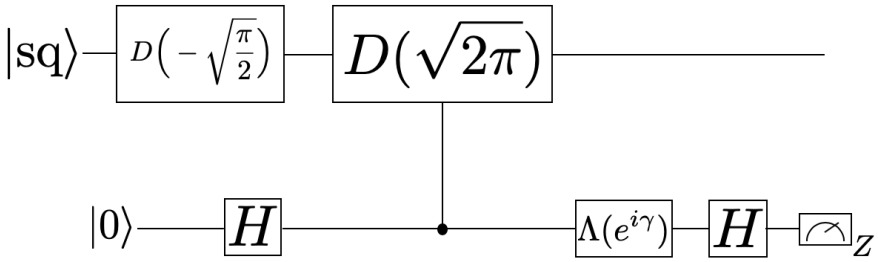


Figure 4.7: Phase estimation circuit for preparing an approximate $|\tilde{0}\rangle$ GKP state. The initial state of the cavity is taken to be the squeezed vacuum state defined in Eq. (4.9). The circuit effectively projects the input squeezed vacuum state onto an approximate eigenstate of the S_p operator while at the same time estimating the phase of the eigenvalue. The phase can be computed so that the output state can be shifted back to an approximate $+1$ eigenstate of S_p.

Given the final measurement record $x[M]$, the estimated phase $\tilde{\theta}$ is chosen as

$$\tilde{\theta} = \arg \int_{-\pi}^{\pi} d\theta e^{i\theta} P(x[M]|\theta). \tag{4.19}$$

The shift back correction based on the estimated phase $\tilde{\theta}$ is given by $e^{i\frac{\tilde{\theta}}{2\sqrt{\pi}}q}$. Note that for either the adaptive or non-adaptive protocol, the estimated phase and probabilities are computed using Eqs. (4.18) and (4.19).

4.6.2 $(1, \tilde{\delta})$ fault-tolerant state preparation using phase estimation

From the circuit of Fig. 4.7 and from Eq. (4.19), a measurement readout error can result in the wrong estimated phase which in turn results in an incorrect shift-back correction (application of e^{ivq} where v is computed using Eqs. (4.17) and (4.18)). Now suppose that the output state is afflicted by a shift error of the form e^{-iwq}. In this case, the final output state will have a total shift error of the form $e^{-i(v-w)q}$. If $|v-w| \mod \sqrt{\pi} > \sqrt{\pi}/6$, then the shift error will be uncorrectable and thus the phase estimation protocol will not be fault-tolerant [4]. In what follows we will identify an output state of the phase estimation protocol as $x[m]$ if it arises from the measurement outcomes $x_1, \cdots, x_m$ in the fault-free case.

4. Note that $|v - w|$ should be taken modulo $\sqrt{\pi}$ since a perfect $|\overline{0}\rangle$ GKP state in p space has peaks at any integer multiple of $\sqrt{\pi}$.

Consider the case where the phase estimation protocol is implemented in m rounds and let us assume that a single measurement readout error occurs and that all other operations are fault free. In this case the output state $x_{\text{out}}[m]$ will have one bit which differs from the bit string $x_{\text{corr}}[m]$ that would have been obtained in the fault free case (so that the Hamming distance $d_H(x_{\text{out}}[m], x_{\text{corr}}[m]) = 1$). The shift-back operation of Eq. (4.17) applied to the output state will be the shift correction of an output state that is one Hamming distance away from the actual state that was produced. Thus to ensure that the protocol is $(1, \tilde{\delta})$ fault-tolerant for some $\tilde{\delta}$, we should only accept output states $x_{\text{out}}[m]$ which have the property that applying the shift back correction corresponding to *any* other state $x'[m]$ with $d_H(x'[m], x[m]) = 1$ results in a correctable left-over shift error. These remarks motivate the following protocol to prepare an approximate $|\tilde{0}\rangle$ state which is fault-tolerant to a single measurement readout error.

Calculation of acceptance set A_m and $\tilde{\delta}$.

Consider all output states obtained from an m round *fault-free* phase estimation protocol of Section 4.6.1. Let $A_m = \emptyset$ (which we call the acceptance set) and $\Gamma_m = \emptyset$. For each output state $x[m]$, do the following

1. Compute the shift correction $v_m = \frac{1}{2\sqrt{\pi}} \arg \int_{-\pi}^{\pi} d\theta e^{i\theta} P(x[m]|\theta)$.
2. Let $j[m]$ be a bit string of size m. For all $j \in \{1, \cdots, m\}$ such that $d_H(x[m], j[m]) = 1$, compute the shift correction $v_m^{(j)} = \frac{1}{2\sqrt{\pi}} \arg \int_{-\pi}^{\pi} d\theta e^{i\theta} P(j[m]|\theta)$.
3. If $\nexists\ j$ such that $|v_m - v_m^{(j)}| \mod \sqrt{\pi} > \frac{\sqrt{\pi}}{6}$, append $x[m]$ to A_m and $\forall\, j \in \{1, \cdots, m\}$, append $|v_m - v_m^{(j)}|$ to Γ_m.

   If $A_m \neq \emptyset$, $\tilde{\delta} = \max_m \Gamma_m$.

$(1, \delta)$ fault-tolerant m round phase estimation protocol for measurement readout errors.

Consider the acceptance set $A_m \neq \emptyset$ and $\tilde{\delta}$ computed from the procedure described above. During an implementation of the phase estimation protocol subject to measurement readout errors, if the obtained output state $x[m] \notin A_m$, abort the protocol and start anew.

Note that during the protocol, there can be more than one measurement readout error. However if more than one readout error occurs, it is possible that the output state is afflicted by a shift error of magnitude greater than $\sqrt{\pi}/6$. Our protocol only guarantees protection against a single readout error.

Four round PE protocol acceptance set A_4	1111	1010	0101	1000	0010
Largest shift difference	0.235	0.225	0.225	0.225	0.225
Probability of obtaining output state	0.1258	0.1287	0.1279	0.0505	0.0499

Table 4.1: The first row displays the accepted measurement strings (set A_4) for the four round non-adaptive phase estimation protocol. If the protocol does not output an element of A_4, the output is rejected and the protocol begins anew. The second row displays the largest shift difference that can occur when applying the phase estimated shift in the presence of a single measurement readout error. As can be seen, these are all less than $\sqrt{\pi}/6 \approx 0.295$. The third row gives the probabilities of obtaining each state at the output of the phase estimation protocol.

As an example, consider the four round phase estimation protocol. Applying the procedure described above for the adaptive phase estimation protocol, we find that A_4 is empty when choosing the initial phase to be $\gamma_0 = 0$ or $\gamma_0 = \frac{\pi}{2}$. This indicates that the adaptive phase estimation protocol of four rounds described in Section 4.6.1 is *not* fault-tolerant to single measurement readout errors[5]. If the phases are computed using the non-adaptive phase estimation protocol and

5. However it is possible that the adaptive phase estimation protocol described in Section 4.6.1 could be modified to be fault-tolerant to measurement readout errors. We leave such analysis to future work.

applying the protocol described above, we find that $\tilde{\delta} = \frac{\sqrt{\pi}}{6} - 0.235 \approx 0.0604$. The set A_4 of accepted measurement strings is given in Table 4.1 and has $|A_4| = 5$. The total probability of obtaining a state in A_4 is roughly 48.3%. For the phase estimation protocol with more than four rounds, the acceptance set and the acceptance probability is very sensitive to the initial phase γ_0 and the domain of the optimized γ. For example, if we choose the initial phase γ_0 to be $\frac{\pi}{2}$ and the range of γ to be $[0, 2\pi]$, we get 18 accepted states (for the adaptive protocol). In this case the total probability of obtaining a state in A_8 is 6.25%. If we choose the initial phase γ_0 to be 0, the acceptance set has four states and the probability of obtaining a state in A_8 is 1.3%. For the non-adaptive phase estimation protocol, $|A_8| = 66$ but the total probability of obtaining a state in A_8 is roughly 79.2%. These results indicate that although the adaptive phase estimation protocol outperforms the non-adaptive protocol in the fault-free case ([179, 44]), the adaptive phase estimation protocol cannot be used in the presence of measurement readout errors for four rounds, and has significantly fewer states in A_8 for eight rounds. Therefore in a noisy implementation of phase estimation, the non-adaptive protocol is preferable. A list of the states belonging to the acceptance set A_8 for both the adaptive and non-adaptive protocols can be found at https://github.com/godott/GKP$_{p}hase_{e}stimation.git.Supposenowthatfor\mathrm{m}roundsthesetA_{m}$ is not empty. We then know there exists a $\tilde{\delta}$ such that the protocol is a $(1, \tilde{\delta})$ fault-tolerant m round phase estimation protocol for *measurement* readout errors. However there are other fault locations where a single fault resulting in a qubit error could potentially lead to an uncorrectable shift error on output states. Note that an X error prior to applying the controlled-$D(\sqrt{2\pi})$ gate will do nothing since the qubit is in the $|+\rangle$ state. A Z error will just change the sign of one of the peaks of the output state in p space. However a damping event occurring on the ancilla qubit before or *during* the application of the controlled-$D(\sqrt{2\pi})$ gate can result in incorrectly applying the $D(\sqrt{2\pi})$ displacement on the cavity. If several damping events occur during the protocol, the output state in p space could potentially be badly deformed. In [179] it was proposed to replace the ancilla qubit by a k-qubit cat state so that if a single qubit undergoes damping, a single shift error of size $D(\frac{\sqrt{2\pi}}{k})$ would occur which can be made small for large k. However this approach would require

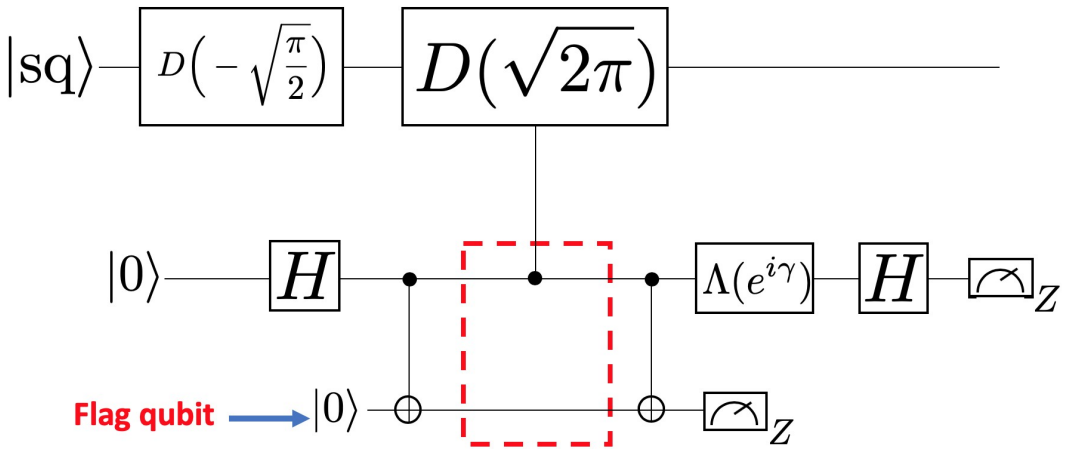


Figure 4.8: Phase estimation circuit with an additional flag qubit. If a damping event occurs during the application of the controlled-$D(\sqrt{2\pi})$ gate resulting in a $D(\sqrt{2\pi})$ error, the flag qubit will be measured as 1 instead of 0. If the flag is measured as 1, we abort the protocol and start anew.

the fault-tolerant preparation of a large k-qubit cat state which would significantly increase the required resources in addition to adding substantially more locations where errors can occur. Similar issues were observed in [153] for preparing cat codes. However large cavity displacements were mitigated by interacting the cavity with a three level system $|f\rangle$, $|e\rangle$ and $|g\rangle$ instead of a qubit. A transition from $|f\rangle$ to $|e\rangle$ would not cause the incorrect gate from being applied. Thus two damping events would be required to cause the incorrect gate from being applied.

Here we propose an alternative solution for mitigating damping errors during the application of the controlled-$D(\sqrt{2\pi})$ gate which uses a single extra flag qubit [25, 24, 19, 178, 151, 20]. Consider the circuit in Fig. 4.8. If a bit flip error occurs before or during the application of the controlled-$D(\sqrt{2\pi})$ gate, the flag qubit will be measured as 1 instead of 0. In such a case the strategy will be simply to abort the protocol and start anew. The analysis for a general damping event using the flag qubit is given in Section B.1. It is shown that if ever a damping event on the $|+\rangle$ ancilla qubit causes the wrong $D(\sqrt{2\pi})$ gate to be applied, the flag qubit will be measured as 1 instead of 0 in which case we abort the protocol and start anew. Thus with the flag qubit, the phase estimation protocol is robust to a damping error during the application of the controlled displacement gate. Note however that if a fault occurs in addition to a damping event, the shift error resulting from the damping event can potentially go undetected. For instance, if in addition

to damping, the flag qubit is subject to a measurement error, the measurement outcome can be 0 instead 1 resulting in acceptance when the protocol should have been aborted.

Suppose that only a single damping event occurs during the four round phase estimation protocol, and that all other operations are fault-free. Using the analytic expressions derived in Section B.1 and for a damping rate of $p = 0.5$, we performed simulations to numerically characterize the effects of qubit damping on the prepared GKP state. We found that the shift error resulting from a single damping event was negligible compared to the largest shift error resulting from a single measurement readout error.

Lastly, an X error prior to the $\Lambda(e^{i\gamma})$ gate will change the sign of γ. After propagating through the Hadamard gate, the X error will become a Z error which will not affect the measurement outcome of the ancilla qubit. For the four round non-adaptive phase estimation protocol, we performed a simulation which showed that the shift error resulting from a single X error prior to the $\Lambda(e^{i\gamma})$ gate is much smaller than the largest shift error arising from a measurement readout error on the ancilla qubit.

Let us assume that the noise model afflicting the oscillator can cause a shift error of size at most $\tilde{\delta}$ in p space (i.e. a shift of the form $e^{-i\tilde{\delta}q}$). From the above, the largest shift error that can arise from a single fault afflicting the qubit space during the four round fault-tolerant phase estimation protocol is $0.235 < \sqrt{\pi}/6$. Following Definition 14, we conclude that the protocol described in this section is a $(1, \tilde{\delta})$-fault-tolerant state preparation of an approximate logical $|\overline{0}\rangle$ GKP state with $\tilde{\delta} = \sqrt{\pi}/6 - 0.235 \approx 0.06$.

In Section 4.7 we will discuss a physical implementation of the phase estimation protocol using circuit QED. A much more detailed analysis of the noise afflicting the cavity and the resulting shift errors will be provided.

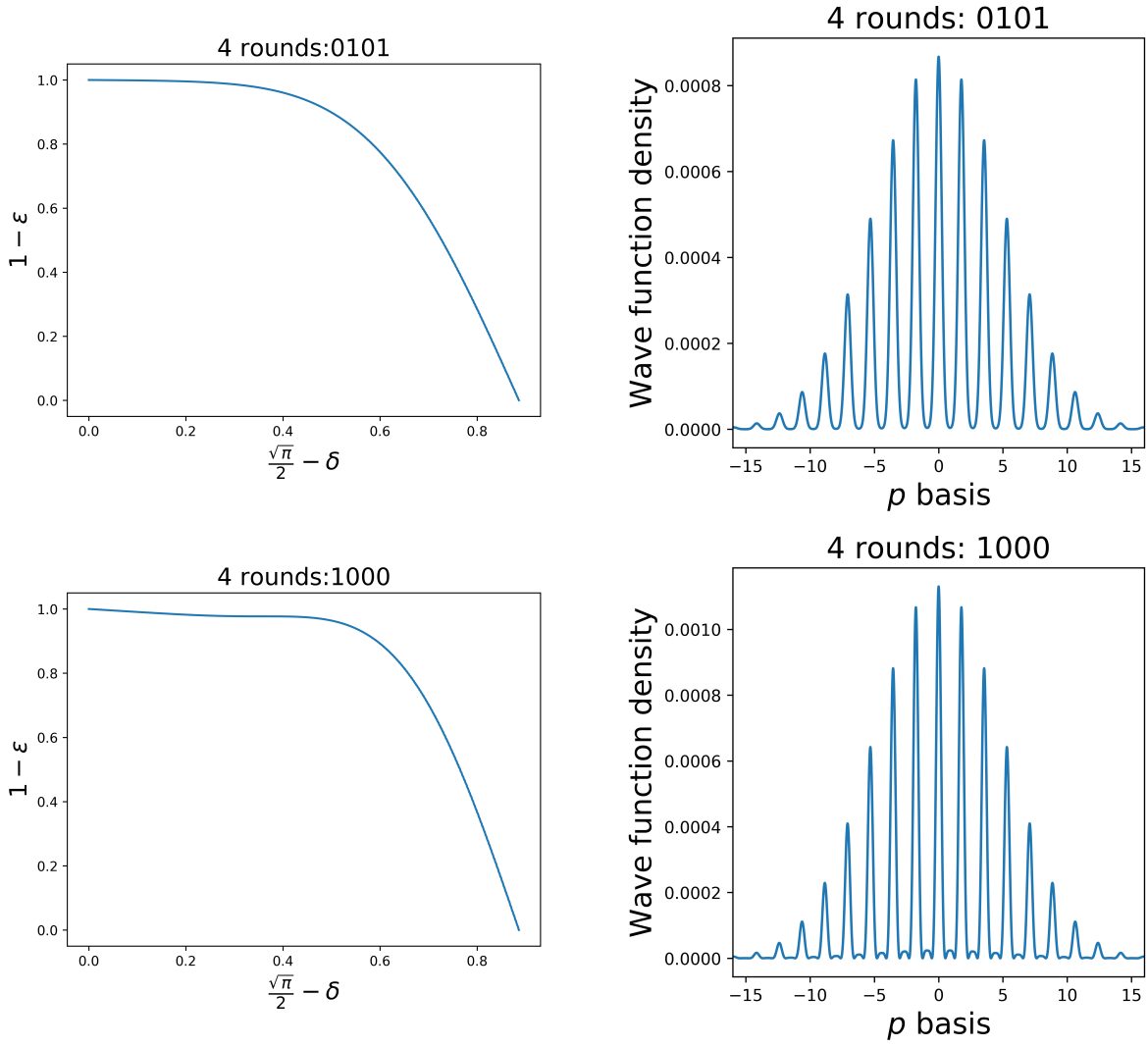


Figure 4.9: (LEFT) Plot illustrating the probability of correcting a shift error of size $\frac{\sqrt{\pi}}{2} - \delta$ for the states 0101 and 1000 obtained via a non-adaptive noise free simulation of the phase estimation protocol presented in Section 4.6. (RIGHT) Wave function density $|\psi(p)|^2$ of the states 0101 and 1000 illustrated in p space. The horizontal axis corresponds to values of p.

4.6.3 Goodness of approximate $|\tilde{0}\rangle$ states obtained from the noise free phase estimation protocol

In Section 4.7.2, the fault tolerant implementation of phase estimation described in this section is analyzed for a noise model which introduces gate noise, measurement readout errors and ancilla state preparation errors in addition to photon loss, amplitude damping and dephasing. Here we consider a noiseless implementation of the four round non-adaptive phase estimation protocol described in this section which will be used to benchmark the noisy implementation.

Since the shift correction obtained from non-adaptive phase estimation is in p space, for a range of values for δ, we computed ϵ using the integral in Definition 10 after performing a shift back correction using Eq. (4.16). This allowed us to compute the probability of correcting shift errors in p of size $\frac{\sqrt{\pi}}{2} - \delta$. Plots for the states 0101 and 1000 which belong to A_4 (see Table 4.1) are given in Fig. 4.9. The $(\frac{\sqrt{\pi}}{2} - \delta, \epsilon)_p$ correctable properties of the other states in A_4 are similar to the ones shown in Fig. 4.9. It can be seen that for small values of size $\frac{\sqrt{\pi}}{2} - \delta$, both states can correct the shift error with probability close to 1.

4.7 Circuit QED Implementation

4.7.1 State evolution in the dispersive regime

In this section we will describe a direct implementation of the controlled-$D(\sqrt{2\pi})$ gate. From [55, 14], the Hamiltonian describing the coupling between a qubit and a cavity in the dispersive regime is given by

$$H_s = \tilde{\omega}_r a^\dagger a + \tilde{\omega}_a Z + \chi a^\dagger a Z - \phi (a^\dagger a)^2 Z, \tag{4.20}$$

where $\phi = \frac{g^4}{\Delta^3}$, $\chi = \frac{g^2}{\Delta} - \phi$, $\tilde{\omega}_r = \omega_r + \phi$ and $\tilde{\omega}_a = \frac{\omega_a + \chi}{2}$. Here g is the coupling strength between the qubit and the cavity and $\Delta = |\omega_r - \omega_a|$. The term $\phi(a^\dagger a)^2 Z$ corresponds to the

non-linear dispersive shift. The Hamiltonian in Eq. (4.20) can be derived by performing an exact diagonalization of the Jaynes-Cummings Hamiltonian and keeping only leading order terms in ϕ [14]. Note that a more systematic treatment of the qubit as an anharmonic oscillator leads to an additional term in Eq. (4.20) given by $-\frac{K}{2}(a^\dagger a)^2$ which is referred to as the Kerr non-linearity [136]. Hence, in our analysis, we choose the system Hamiltonian to be given by

$$H_s = \tilde{\omega}_r a^\dagger a + \tilde{\omega}_a Z + \chi a^\dagger a Z - \phi(a^\dagger a)^2 Z - \frac{K}{2}(a^\dagger a)^2. \tag{4.21}$$

The direct implementation of the controlled-$D(\sqrt{2\pi})$ gate can be achieved using the drive Hamiltonian

$$H_d(t) = \mathcal{E}(t) a^\dagger e^{-i\omega_d t} + \mathcal{E}^*(t) a e^{i\omega_d t}, \tag{4.22}$$

where $\mathcal{E}(t)$ describes the pulse shape of the drive and ω_d is the drive frequency. Thus the total Hamiltonian describing the evolution of the qubit-cavity system during the implementation of the controlled-$D(\sqrt{2\pi})$ gate is given by

$$H(t) = H_s + H_d(t). \tag{4.23}$$

Going into the rotating frame of the qubit and the cavity and defining $\phi_\pm = \phi \pm K$ and $\omega_\pm = \pm\chi$, in Section B.2 we show that

$$\begin{aligned} V_R(0,T) &= \mathcal{T} e^{-i\int_0^T H(t')dt'} \\ &= R_+(T) D(A_+) e^{iB_+} |0\rangle\langle 0| + R_-(T) D(A_-) e^{iB_-} |1\rangle\langle 1|, \end{aligned} \tag{4.24}$$

where

$$R_{\pm}(T) = e^{-iT\omega_{\pm}a^{\dagger}a}(1 \pm iT\phi_{\pm}(a^{\dagger}a)^2), \tag{4.25}$$

$$A_{\pm} = -i\int_0^T \mathcal{E}(t)e^{i\omega_{\pm}t}dt \mp \phi_{\pm}(2a^{\dagger}a - 1)\int_0^T \mathcal{E}(t)te^{i\omega_{\pm}t}dt + \mathcal{O}(\phi_{\pm}^2), \tag{4.26}$$

$$\begin{aligned} B_{\pm} = &-\frac{i}{2}\int_0^T dt\int_t^T dt'\Big(\mathcal{E}^*(t')\mathcal{E}(t)e^{i\omega_{\pm}(t'-t)} - \mathcal{E}(t')\mathcal{E}^*(t)e^{-i\omega_{\pm}(t'-t)}\Big) \\ &\pm \frac{\phi_{\pm}(2a^{\dagger}a+1)}{2}\int_0^T dt\int_t^T dt'\Big(\mathcal{E}^*(t')\mathcal{E}(t)e^{i\omega_{\pm}(t'-t)} + \mathcal{E}(t')\mathcal{E}^*(t)e^{i\omega_{\pm}(t'-t)}\Big)(t'-t) + \mathcal{O}(\phi_{\pm}^2). \end{aligned} \tag{4.27}$$

In deriving Eqs. (4.25) to (4.27), we kept only leading order terms in $\phi_{\pm}$ since the Hamiltonian in Eq. (4.21) neglects higher order terms in ϕ and K. We keep only leading order terms since with current experimental parameter values, $\phi_{\pm}$ is roughly three orders of magnitude smaller than χ.

Firstly, notice that the terms $R_{\pm}(T)$ introduce a relative rotation between the $|0\rangle$ and $|1\rangle$ state of the qubit. Neglecting the terms proportional to $\phi_{\pm}$, it was shown in [179] that by choosing the total interaction time to be $T = \pi/\chi$, the relative rotation between $|0\rangle$ and $|1\rangle$ can be set to one. However due to the presence of the non-linear dispersive shift and Kerr terms in Eq. (4.25), it is clear that the relative rotation cannot be completely removed. Further, $R_{\pm}(T)$ does not depend on the pulse shape $\mathcal{E}(t)$ of the drive term. Therefore even with an optimized pulse shape (see below), the effects from the non-linear dispersive shift and the Kerr will cause an undesired relative rotation between the qubit states. Regardless, we will still choose the interaction time T to be π/χ to ensure that we eliminate the relative rotation from the leading order terms in Eq. (4.25). In Section B.2.4, we provide further details using the analytic expressions to show that to mitigate the effects due to the non-linear dispersive shift and Kerr terms would require the protocol to take place on time

scales of order $\frac{1}{\phi_\pm}$ which (for the parameter values in Table 4.2) is four orders of magnitude larger than $\frac{\pi}{\chi}$. For such long time scales, noise terms such as photon loss, dephasing and damping would render the protocol impractical.

The terms in Eq. (4.24) that perform the desired controlled displacement gate are given by $A_\pm$ in Eq. (4.26). The goal is to choose a pulse shape that implements the desired gate while at the same time minimizes the contributions from the non-linear dispersive shift and the Kerr term (terms proportional to $\phi_\pm$ in Eq. (4.26)). In order to be experimentally relevant, it is important to choose a pulse shape that is accessible to near term experiments using 2D and 3D cavities. We chose a Gaussian pulse with the following parameters

$$\mathcal{E}(t) \approx -2.09562\Big(\frac{\sqrt{2\pi}}{T}\Big)e^{-\frac{(t-\mu)^2}{2\sigma^2}}, \tag{4.28}$$

where $\mu = \frac{\pi}{2\chi}$ and $\sigma = \frac{\pi}{10\chi}$. With this Gaussian pulse we obtain

$$A_\pm \approx \mp\frac{\sqrt{2\pi}}{2} \mp \phi_\pm(2a^\dagger a - 1)\int_0^T \mathcal{E}(t)te^{i\omega_\pm t}dt. \tag{4.29}$$

The amplitude of the Gaussian, given by $-2.09562\Big(\frac{\sqrt{2\pi}}{T}\Big)$, is chosen numerically to ensure that the appropriate gate is being performed.

Since both the A_+ and A_- terms are symmetric with opposite signs, the gate $D\Big(-\sqrt{\frac{\pi}{2}}\Big)$ in Fig. 4.8 is *not* required. Furthermore, the first term in Eq. (4.29) is independent of χ. However the contribution from the second term in Eq. (4.29) will depend on the coupling strength and relative frequencies between the qubit and the cavity. Thus reducing the value of χ will minimize the undesired effects arising from the non-linear dispersive shift and the Kerr term while still producing the desired gate.

The $B_\pm$ terms of Eq. (4.27) result in a phase difference between the $|0\rangle$ and $|1\rangle$ qubit states. However, computing the first integrals in Eq. (4.27) (i.e. terms before $\phi_\pm$), we find that for our chosen pulse, the phase difference remains *fixed* during every round of the phase estimation protocol. Therefore after applying the controlled displacement gate, we apply an additional phase gate

Parameter values	Simulation 1	Simulation 2
p	5×10^{-3}	10^{-3}
$\frac{\kappa}{2\pi}$	1.59×10^{-5}MHz	1.59×10^{-6}MHz
$\frac{\gamma_1}{2\pi}$	1.06×10^{-3}MHz	1.06×10^{-4}MHz
$\frac{\gamma_\phi}{2\pi}$	7.96×10^{-4}MHz	7.96×10^{-5}MHz
g	8.92MHz	5.09MHz
Δ	0.32GHz	0.32GHz
K (Kerr)	10^{-4}MHz	10^{-5}MHz

Table 4.2: Parameter values for the two simulations of the non-adaptive noisy phase estimation protocol. The values for κ are chosen based on state of the art 3D cavities. The parameters in the second column results in a T_1 time given by $T_1 = \frac{2\pi}{\kappa} = 10$ms. For a resonator frequency $f_r = 7$GHz, the quality factor is given by $Q = \frac{f_r}{\kappa/(2\pi)} = 7 \times 10^7$. For all simulations, the squeezing level of the input squeezed vacuum state is chosen to be 0.2.

which removes the phases introduced by the left most integrals of $B_\pm$. Note however that the $\phi_\pm$ terms in $B_\pm$ will not be cancelled and will thus introduce additional shift errors.

4.7.2 Full noise analysis and master equation results.

In this section we perform a numerical analysis for the noisy implementation of the phase estimation protocol described in Section 4.6.2. The simulation is performed in several steps that we now describe. First, the controlled displacement gate is modeled using the following master equation

$$\dot{\rho} = -i[H(t), \rho] + \kappa \mathcal{D}[a]\rho + \gamma_1 \mathcal{D}[\sigma_-]\rho + \gamma_\phi \mathcal{D}[\sigma_z]\rho, \tag{4.30}$$

where $H(t)$ is given by Eq. (4.23) after going into the rotating frame and $\mathcal{D}[L]\rho = (2L\rho L^\dagger - L^\dagger L\rho - \rho L^\dagger L)/2$. The density matrix corresponds to the joint state of the cavity, ancilla qubit and flag qubit. The parameters κ, γ_1 and γ_ϕ correspond to the photon loss rate, the qubit decay rate and qubit dephasing. The pulse shape of the drive is given by Eq. (4.28). The total evolution time of the controlled displacement is given by $\frac{\pi}{\chi}$ with $\chi = \frac{g^2}{\Delta} - \phi$.

Both before and after the controlled displacement gate, the state of the cavity is subject to photon loss and its evolution is computed by solving a master equation. Based on current gate, state preparation and measurement times, we chose the evolution time from the preparation of the

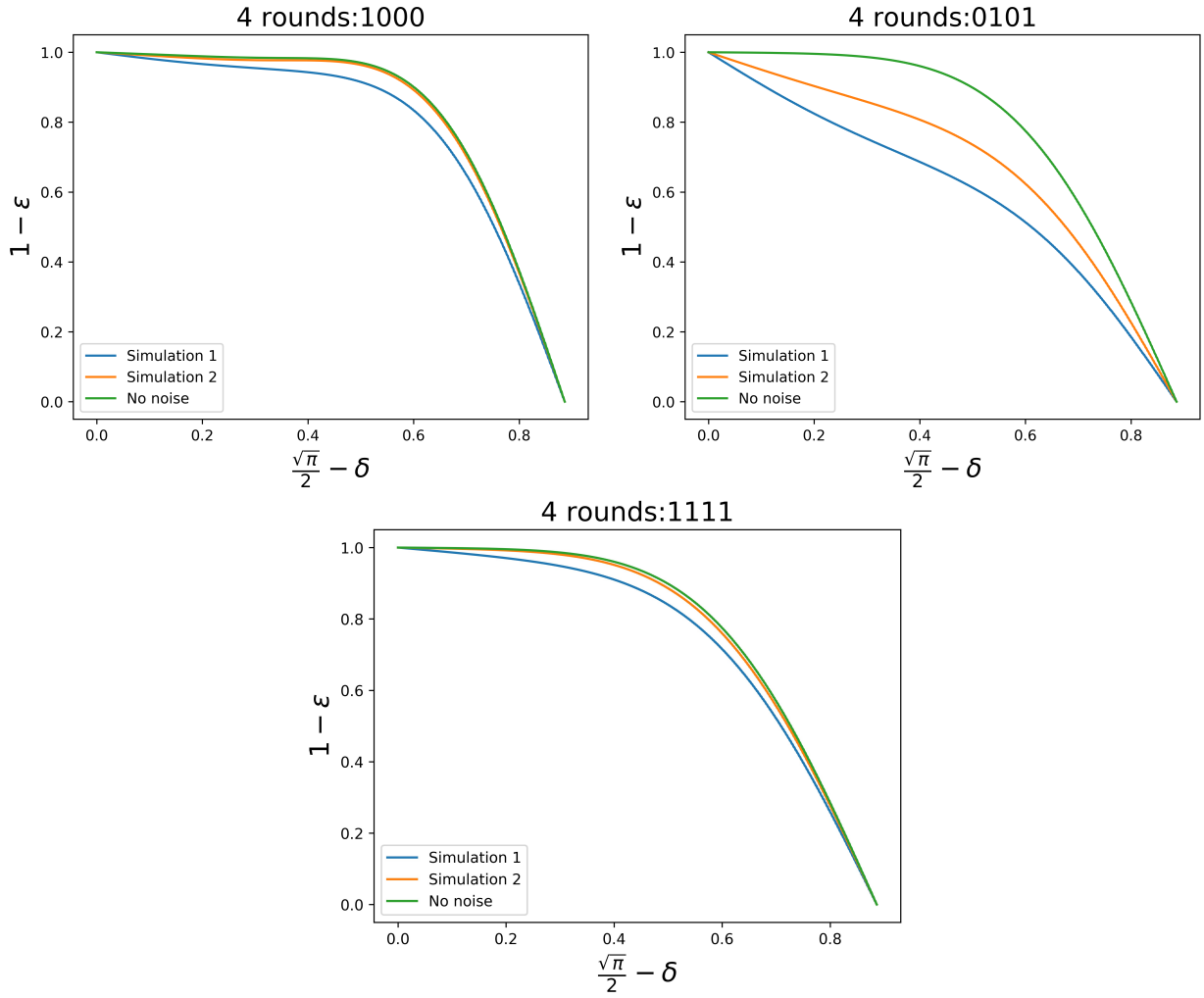


Figure 4.10: Probability of correcting a shift errors of size $\frac{\sqrt{\pi}}{2} - \delta$ for parameters chosen from the second (labeled "Simulation 1") and third (labeled "Simulation 2") column of Table 4.2 in addition to the case where no noise is present. The plots are for the states 1000, 0101 and 1111 obtained from the four round fault-tolerant phase estimation protocol described in Section 4.6.

ancilla to the first CNOT gate (after which the controlled-displacement gate is performed) to be 0.14μs. Similarly, the evolution time after the controlled-displacement gate to the time the ancilla is measured is also chosen to be 0.14μs. Thus during one round, the cavity freely evolves with photon loss for a total of 0.28μs. In addition, before and after the controlled displacement gate, we allow all qubit locations to fail with the following depolarizing noise model

1. With probability p, each two-qubit gate is followed by a two-qubit Pauli error drawn uniformly and independently from $\{I, X, Y, Z\}^{\otimes 2} \setminus \{I \otimes I\}$.

2. With probability $\frac{2p}{3}$, the preparation of the $|0\rangle$ state is replaced by $|1\rangle = X|0\rangle$.

3. With probability $\frac{2p}{3}$, any single qubit measurement has its outcome flipped.

4. With probability $p/10$, each Hadamard gate is followed by a Pauli error drawn uniformly and independently from $\{X, Y, Z\}$.

We chose $p/10$ for Hadamard gate failures since for current superconducting architectures, single qubit gate fidelities are about an order of magnitude higher than two-qubit gate fidelities. In practice, the gate errors applied to the qubit Hilbert space will depend strongly on the circuit QED architecture and should also be modeled using a master equation as in Eq. (4.30). However, we chose a depolarizing model in order to reduce the computation time and simplify the analysis. We also mention that in our analysis we assumed that g is tunable. Thus both before and after the controlled-displacement gate, the cavity and qubit system is decoupled and can be treated separately.

The master equation was numerically solved using Qutip [86], our code can be accessed at https://github.com/godott/GKP_phase_estimation.git. Due to the long computation time during the controlled displacement gate, performing a full Monte Carlo simulation to take into account gate errors with the depolarizing model was unfeasible (i.e. gate locations both before and after the controlled displacement gate). Instead, we analytically computed the error probabilities for each Pauli operator (using the depolarizing noise model described above) immediately after the first

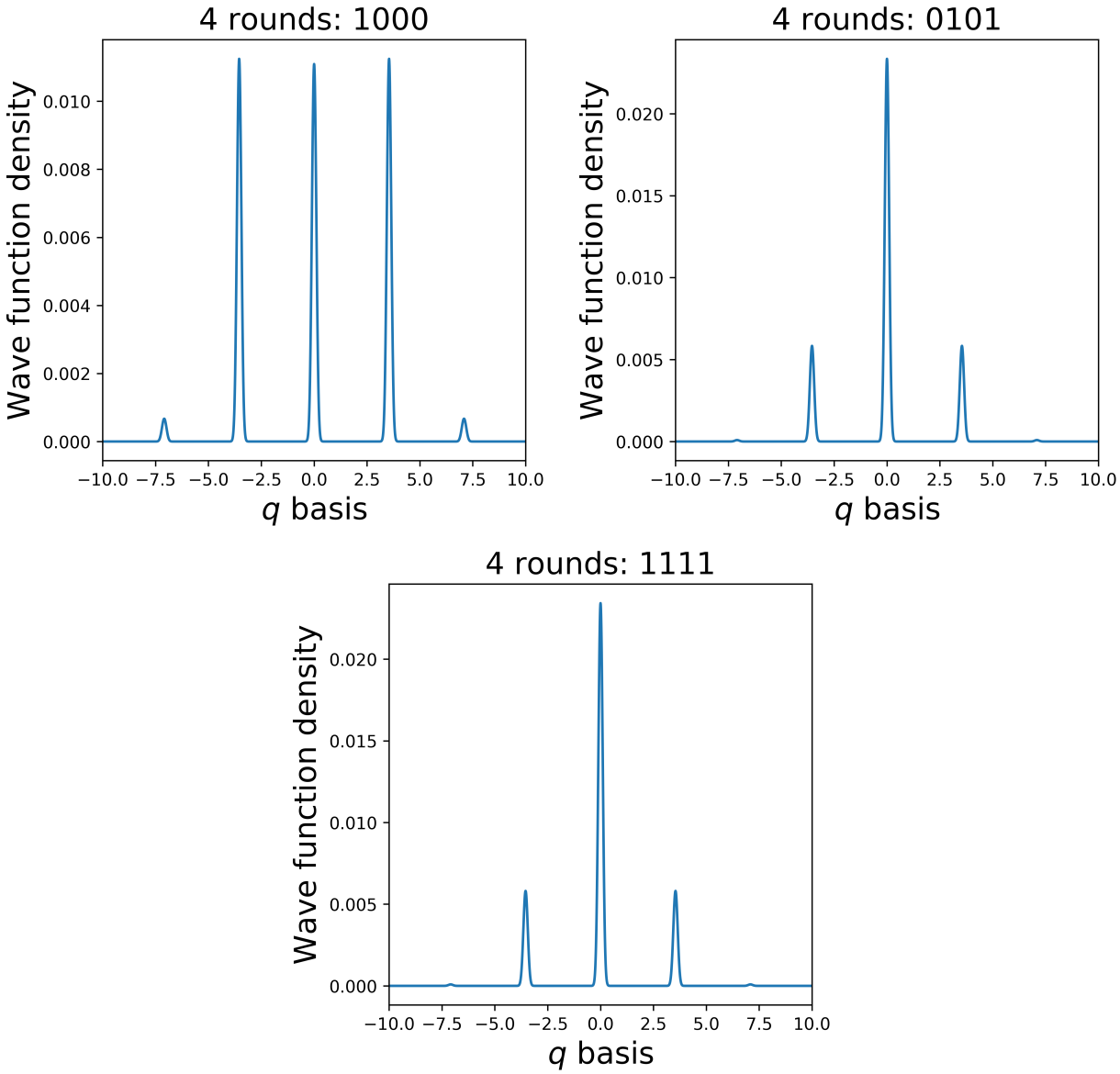


Figure 4.11: Wave function density $|\psi(q)|^2$ of the states 1000, 0101 and 1111 obtained from the noise free non-adaptive phase estimation protocol.

Four round PE protocol acceptance set A_4	1111	1010	0101	1000	0010
Probability of output state noisy simulation 1	0.1297	0.0727	0.0737	0.0474	0.0429
Probability of output state noisy simulation 2	0.1468	0.0982	0.0978	0.0514	0.0448

Table 4.3: The second row corresponds to the probabilities of obtaining the output states of A_4 for the parameters chosen from the second column of Table 4.2 conditioned on all flag measurements being 0. The third row is identical but for the parameters chosen from the third column of Table 4.2. The probabilities are smaller for noisier circuits since the flag qubit has a higher chance of being measured as 1 causing the protocol to be aborted.

CNOT gate, before both measurement locations and before the phase gate[6]. At each location, all possible Pauli operators based on their associated probabilities were added. For a given Pauli error, the probabilities (which are expressed as functions of p) were then used (in addition to the state evolution obtained from the master equation) to compute the final probability of obtaining a given output state. Instances with two or more faults occurring on the qubit Hilbert space before and after the controlled-displacement gate were neglected. However our analysis is still more complete than previous implementations which considered only measurement errors and errors during the controlled-displacement gate. More details of the Pauli simulation can be found in Section B.2.5.

We performed two different simulations where for each simulation, the chosen parameter values are given in Table 4.2. The parameters chosen for the first simulation (middle column in Table 4.2) are based on current experimental values for 2D and 3D cavities [27, 153, 56, 149, 121, 16]. The parameters chosen for the second simulation are based on values that might be obtained with improved future technologies. Plots showing the probability of correcting shift errors $\frac{\sqrt{\pi}}{2} - \delta$ after performing a shift correction of the output states of the noisy phase estimation protocol using Eq. (4.16) are given in Fig. 4.10. In what follows we will refer to these plots as $\epsilon - \delta$ plots. Note that noise during the phase estimation protocol introduces more shift errors in p space. Therefore in Fig. 4.10, only $\epsilon - \delta$ plots in p space are shown.

For the four round phase estimation protocol, we find that the state 1010 has a similar $\epsilon - \delta$ plot to the one for the state 0101 whereas the state 0010 has a similar $\epsilon - \delta$ plot to the one for the

6. The error probabilities were computed by considering all possible single-fault locations resulting in a given error at the considered location. For instance, a $Z \otimes I$ error after the first CNOT gate can arise from a $Z \otimes I$ error from the faulty CNOT gate, but also from a Z error after the application of the Hadamard gate prior to the CNOT gate.

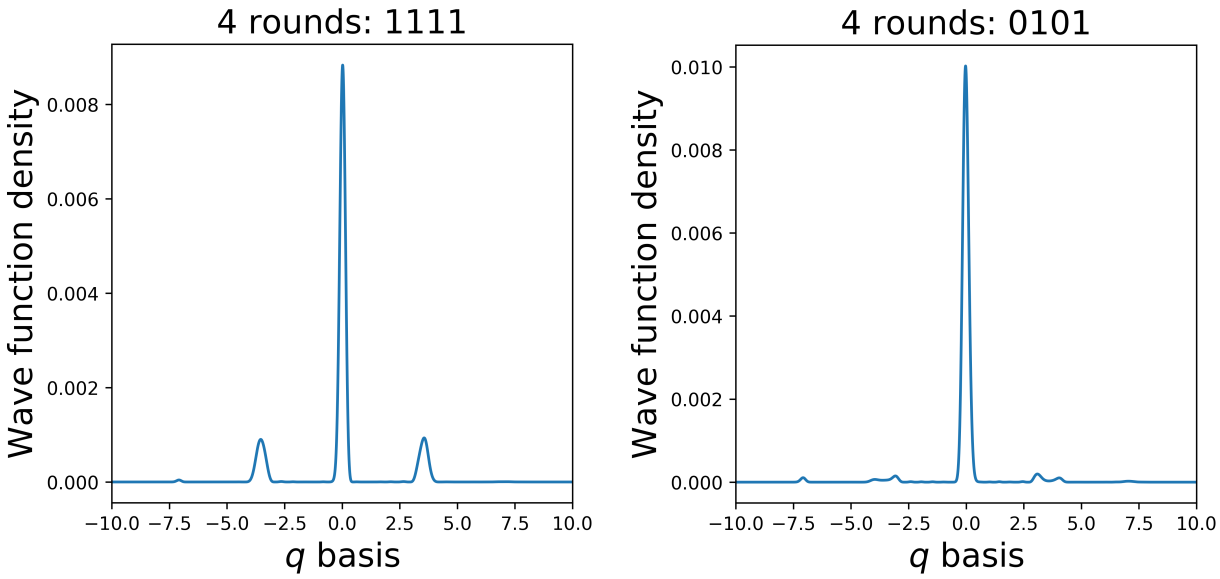


Figure 4.12: Wave function density of the states 1111 and 0101 where all noise terms are excluded apart from the Kerr and non-linear dispersive shift terms. Comparing to Fig. 4.11, the Kerr and non-linear dispersive shift terms reduce the amplitudes of the two small peaks of the 0101 state significantly more than those for the 1111 state.

state 1000. The probability of obtaining each state for the two noisy simulations (parameters in Table 4.2) are given in Table 4.3. It can be seen that the probability of the state 0101 to correct shift errors is significantly more affected by the noise than the state 1000 or 1111. To understand why this is the case, it is useful to look at the wave function densities for these three states in the q basis when no noise is present (see Fig. 4.11). Comparing the wave function densities, it can be observed that the state 1000 has three peaks with similar amplitudes, whereas the states 0101 and 1111 have two smaller peaks compared to the peak at the center. Performing a numerical analysis, it is found that when only damping is present (with all other noise terms including Kerr and non-linear dispersive shift set to zero), damping has a negligible effect on the height of the peaks for all considered states. However performing a numerical simulation with only the Kerr and non-linear dispersive shift terms present, these terms significantly reduce the height of the peaks for the state 0101 and 1010. Therefore, these states are left with one large peak in q with all other peaks close to zero which results in a wave function density with very low resolution in p space (which thus affects the ability for these states to correct shift errors in p). Interestingly, the state 1111 is

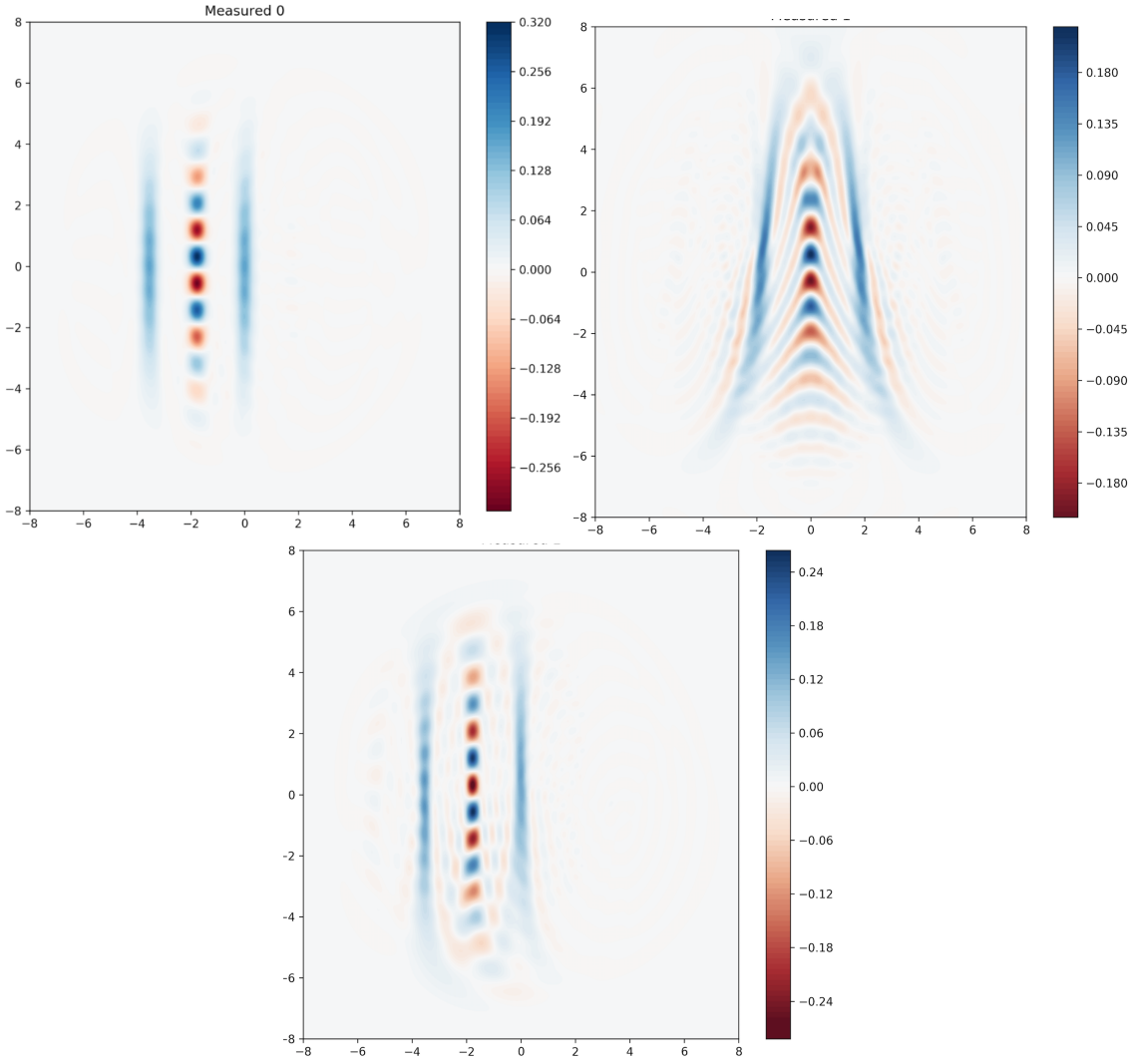


Figure 4.13: Comparison of the state prepared by one round of the PE protocol with different levels of Kerr (Kerr=$0MHz, 1^{-4}MHz, 1^{-}3MHz$.

more robust to the Kerr and non-linear dispersive shift contributions since there is a much smaller reduction in the amplitudes of the smaller peaks compared to those for the states 0101 and 1010 (see Fig. 4.12). Furthermore, the states 1000 and 0010 have three large peaks in q and thus even with a reduced amplitude, these states have a higher wave function density resolution in p space.

Lastly, to address the impact of the Kerr non-linearity, we include a comparison of one round of the PE protocol with different levels of Kerr in Fig. 4.13.

CHAPTER 5

THE CERTIQ VERIFICATION FRAMEWORK

5.1 The Need and Challenges of Formal Verification in Quantum Compilation

As introduced in Chapter 3, quantum compiler is an essential component in the quantum software stack, bridging quantum hardware and useful applications. In the near-term, a quantum compiler must perform heavy optimizations on quantum programs to fit programs onto quantum devices of limited qubit lifetime and connectivity. This makes quantum compiler code error-prone, as writing correct quantum compilation transformations can be complicated. Bugs in quantum compilers will corrupt the execution of programs and can lead to misleading results in scientific research performed on the quantum computers (QCs). In the open-source standard, Qiskit, compiler's issue page [181], there have been numerous bugs reported to date. Undetected bugs can affect the millions of simulations and real quantum machine runs executed by more than 100k users on the Qiskit platform. Thus, eliminating bugs in quantum compilers becomes a crucial problem for the success of near-term quantum computation.

Unfortunately, testing cannot provide a complete solution to the problem of quantum compiler debugging. Testing on real devices is impractical because of the noise in the hardware and the cost of state tomography. Testing using classical simulation is also not scalable since it requires exponential memory/time.

In theory, formal verification provides a solution to this problem by proving that the compiler implementation preserves the program semantics for source programs of any size. For example, in the classical case, the CompCert C compiler [107] is formally verified by constructing machine-checkable proofs in the Coq proof assistant [30]. There have been a few efforts to verify the quantum compilation process by formal methods [78, 10, 166]. These verifiers, however, are not yet practical in real-world scenarios for three key reasons. First, these works have only shown the ability to verify simple compiler optimizations using limited quantum data representations. How-

ever, compiler optimizations in a realistic compiler require more than these limited representations of quantum data. Second, these verifiers are written in proof assistants like Coq and F*. Thus, expertise in proof writing, familiarity with uncommon languages, and many man-hours of coding are required to conduct verification. Third, these approaches are hard to automate, making it infeasible to verify rapidly developing quantum compilers like Qiskit [181] that are heavily dependent on third-party code.

5.2 CertiQ Verification Framework

5.2.1 CertiQ design

In this section, we introduce CertiQ, a verification framework for writing and verifying IBM Qiskit compiler optimizations. To our knowledge, CertiQ is the first effort enabling the verification of real-world quantum compiler transformations in a mostly-automated manner. By "mostly-automated," we mean programmers need to write few to no specifications and annotations to support verification. Though targeted at quantum verification, the high-level workflow of CertiQ is similar to classical verification frameworks such as Alive [111] and Yggdrasil [165]: code contributors write compiler optimizations and few specifications in CertiQ, and the correctness of the code can be automatically verified through the CertiQ verifier based on satisfiability modulo theory (SMT) solvers. Executable and highly-optimized Qiskit compiler passes can then be generated directly from the compiler passes written in CertiQ. This makes CertiQ readily available for the 100k users of the IBM cloud service who perform over tens of millions of simulations and experiments every year.

The design philosophy underpinning CertiQ is motivated by three practical challenges that arise when automating the verification of a real-world quantum compiler. The first challenge is that efficient quantum circuit equivalence testing is still beyond reach. Checking quantum circuit properties usually requires exponential computation time and memory, especially when using the denotational semantics (i.e., the matrix representation) [131]. The key insight behind our solution

is that, although checking general properties of quantum circuits is hard, checking equivalence—the specific property required to verify quantum compilers—can be formulated as an SMT problem for an automatic checking process with a state-of-the-art SMT solver (Z3 [36]). Thus, CertiQ is not characterized by large amounts of computational overhead as the methodology does not depend on the full denotational semantics for verification. Instead, a set of extracted rules about provably correct circuit rewriting is used to verify quantum program transformations. Based on this idea, CertiQ introduces a quantum circuit calculus that enables symbolic representations and executions of quantum circuits and offers a set of rewriting rules to safely reduce gate counts inserted by compiler passes. CertiQ also provides a verified library containing high-level transformation functions over quantum circuits that are proven to preserve semantics. Verification conditions of compiler passes written in CertiQ can be encoded into SMT formulas using the symbolic execution results of the code as well as with theories (or prerequisites) extracted from the rewriting rules and specifications of library functions.

As for the second challenge, quantum compilers greatly benefit from the ability to manipulate data in multiple representations to better facilitate optimizations, but conversions between different representations are complicated, error prone, and hard to reason about. CertiQ adapts the *forward simulation* technique [113] to define the correctness of such conversions and builds a set of verified conversion functions in the CertiQ library. Using this procedure, we identified a critical bug in the Qiskit implementation that converts quantum data representations, which has been confirmed by the Qiskit team.

Finally, it is impractical to write and maintain two versions of the compiler code—the verified version and the executable version. CertiQ solves this problem by designing a uniform Python-like interface to write, specify, verify, and maintain compiler passes in CertiQ, while relying on the CertiQ synthesizer to produce two versions of the code: the symbolic representations of the compiler pass for the verification purpose and the executable Python code that can be integrated into the Qiskit implementation. This solution is based on an observation that quantum compilers are highly domain specific, where many modules and templates can be abstracted to allow reusability. For

example, CertiQ abstracts three different loop invariant templates that can cover most use cases in quantum compiler passes. We have used CertiQ to write and verify 26 (out of 30) compiler passes in Qiskit (version 0.192), showing the capability of our interface and synthesizer.

In the next section, we first make an introduction to the Qiskit Compiler

5.2.2 The Qiskit compiler

The Qiskit compiler (called Terra) is the foundation of the Qiskit framework, upon which other Qiskit components are built. The Qiskit Compiler consists of a set of tools for composing quantum programs at the level of circuits, optimizing them for the constraints of a particular physical quantum processor, and managing the batched execution of experiments on remote-access backends. The optimizations in Qiskit Terra are crucial for successful execution of quantum programs since quantum resources are scarce and qubit coherence time is very limited.

We describe the main components of the Terra compiler (Fig. 5.1), to which the CertiQ framework provides abstract specifications and contracts.

Quantum register A quantum register is a collection of qubits that provides certain functionality in a quantum algorithm. Every qubit lives in a quantum register.

Coupling map and layout Coupling map is the description of the connectivity of qubits on the physical device. It stores the edges of the qubits in a list. For example,
`coup = [[0,1],[1,2],[2,3]]` describes a device of 3 physical qubits with linear connectivity. A layout is a Python dictionary from the virtual qubits in the quantum register to the physical qubits on the device. For example, the implementation in Terra is,

```
class Layout():
  def __init__(self):
    self._p2v = dict() # Physical to virtual qubit map
    self._v2p = dict() # Virtual to physical qubit map
```

QuantumCircuit QuantumCircuit is the class that stores quantum circuits as a series of instructions on classical and quantum registers. It provides an interface to input quantum circuit descrip-

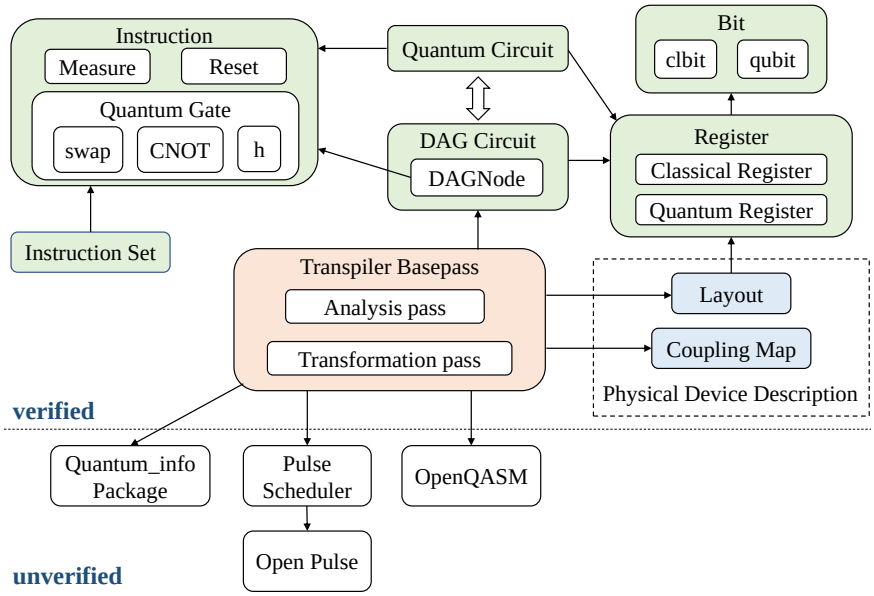


Figure 5.1: Qiskit Terra components and call graph. Green boxes are quantum data structures. Blue boxes are physical devices related data structures. The red box is the transpiler. CertiQ gives specifications to components in blue and green boxes and verifies the transpiler in red box. White boxes under the horizontal dotted-line are parts that are not verified or just partly verified by CertiQ.

tion and for visualization.

DAGCircuit The DAGCircuit class is another description of a quantum circuit and is equivalent to QuantumCircuit. Compared to QuantumCircuit description, DAGCircuit provides more flexible APIs for circuit transformations and to explicitly express the dependence between individual gates in circuits. For example, it provides a method `topological_op_nodes()` that allows users to traverse gates in the DAG in topological sort, easing out lots of circuit optimization algorithms.

Compiler passes The transpiler is the circuit rewriting module in Qiskit Terra responsible for stage 2, 3, and 4 in the quantum compilation process. Because transpiler is the critical part of the compiler and also the fastest iterating component, the need for automated verification is thus pressing. The design language of the transpiler is similar to that of LLVM [105]. It consists of modular components called transpiler passes that can be assembled by the transpiler pass manager with respect to their dependency constraints. Input and output of transpiler passes are both DAGCircuit.

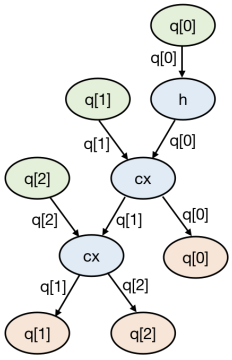
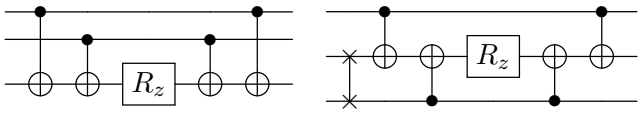


Figure 5.2: The DAGCircuit representation of the circuit that prepares the GHZ state. Green nodes are input nodes, blue nodes are operation nodes and red nodes are output nodes. Arrows represent dependency and are specified by qubit number. DAG representation is equivalent to a quantum circuit description, thus, throughout the paper we will use circuit diagram for visualization for readability.



(a) The un-mapped circuit (b) The mapped circuit

Figure 5.3: An example of circuit transformation made by lookahead swap, one of `routing` passes. The first gate in circuit (b) is a quantum swap gate, which swaps the quantum state of the two connected qubits. Consequently, all the following gates operating on these two qubits have to swap the two operands.

In the Qiskit compiler (version 0.192), there are *seven* types of compiler passes: layout selection, routing, basis change, optimizations, circuit analysis, synthesis, and additional assorted passes. Layout selection passes together with routing passes make sure that the decomposed quantum circuit conforms to the topological constraints of the quantum hardware. The routing procedure is usually done by strategically inserting quantum swap gates (see Fig. 5.3 for an example). Basis change passes help the decomposition of quantum circuits to the gate set supported by the target hardware backend. Optimization passes include various circuit rewriting based optimizations such as gate cancellation [116], scheduling optimization [160], noise adaptation [123], and crosstalk mitigation [125]. Circuit analysis passes do not modify the circuits but return important information about the circuits. Synthesis passes perform large unitary matrix decomposition. Additional passes perform miscellaneous tasks such as circuit validation.

5.3 The CertiQ Workflow

The goal of the CertiQ framework is to allow quantum programmers without much formal verification background to write provably correct quantum transformation passes, which can be readily integrated into the open-source Qiskit compiler and used in real-world quantum experiments.

The CertiQ framework consists of five major components (see Fig. 5.4): (1) A verified library consisting of basic classes of quantum data, high-level equivalent transformation functions over quantum circuits, and conversion functions for different data representations, serving as the basis for building compiler passes in CertiQ (see §5.4); (2) Specifications for the verified library; (3) The quantum circuit calculus (QCC) that introduces the symbolic execution of quantum circuits with a set of pre-proven rules to reduce or rewrite gates inserted by compiler passes (see §5.5); (4) The CertiQ synthesizer, offering a unified interface to write and maintain quantum compiler passes, which can then synthesize the executable Python code (linked with the implementations of the library), as well as the verification conditions (linked with the specification of the library) (see §5.6); (5) A verifier that encodes the verification conditions into SMT formulas and invokes Z3 to solve.

Figure 5.4 shows the workflow of verifying a compiler pass written in CertiQ. In the rest of this section, we will fake a dummy compiler pass as a running example to explain the verification details involving loops, symbolic execution, equivalence rules, and the library. This dummy compiler pass will loop over all the gates in the circuit and, in each iteration, it first inserts two adjacent CX gates and then applies an equivalent transformation T in the library to all the following gates.

The CertiQ synthesizer first statically analyzes the compiler pass code, and produces symbolic representations. The compiler pass in the running example contains a loop and the loop iteration transfers $C; C'$ into $C; CX; CX; T(C')$. The symbolic representation of the circuit after the loop iteration is:

$$\texttt{app}(C; CX; CX; T(C'), Q)$$

which applies the previous circuit C, two adjacent CX gates, and the transformed circuit $T(C')$ to

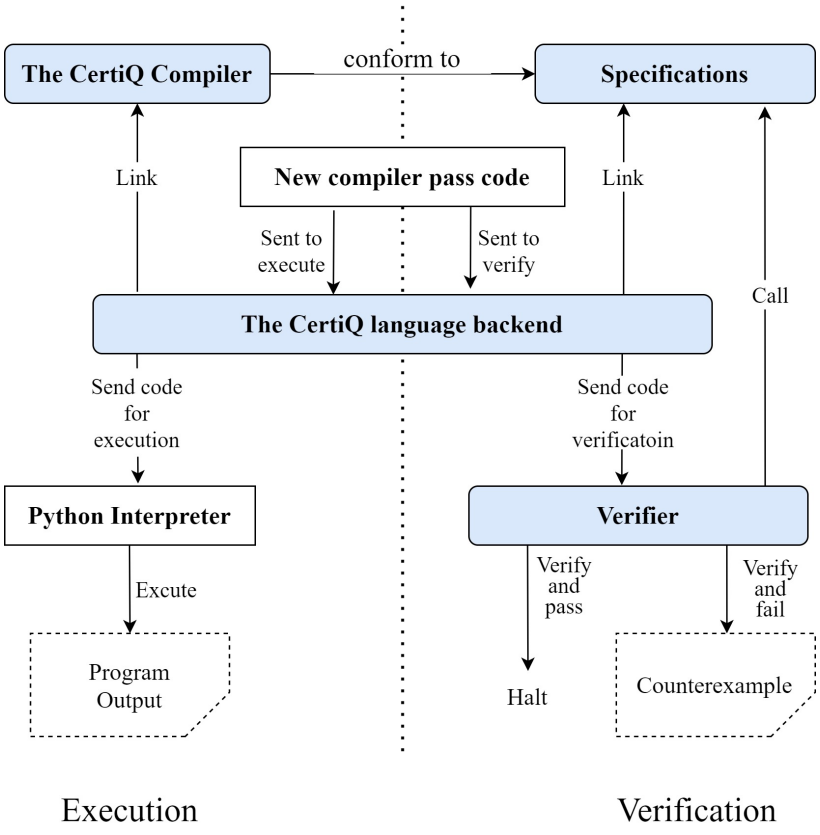


Figure 5.4: CertiQ workflow. Blue boxes are CertiQ components.

an input qubit register Q.

To prove that this pass preserves the semantics, we only need to show that circuits before and after each loop iteration are equivalent, i.e.,

$$\texttt{app}(C; C', Q) \equiv \texttt{app}(C; CX; CX; T(C'), Q)$$

CertiQ then performs the symbolic execution on both sides:

$$\texttt{app}(C; C', Q) \rightharpoonup \texttt{app}\left(C', \texttt{app}(C, Q)\right)$$

$$\begin{aligned}
&\texttt{app}(C; CX; CX; T(C'), Q) \\
&\rightharpoonup \texttt{app}\left(CX; CX; T(C'), \texttt{app}(C, Q)\right) \\
&\dots \\
&\rightharpoonup \texttt{app}\left(T(C'), \texttt{app}(CX, \texttt{app}(CX, \texttt{app}(C, Q)))\right)
\end{aligned}$$

Thus, the proof goal becomes:

$$\mathbf{G} : \texttt{app}\left(C', \texttt{app}(C, Q)\right) \equiv \texttt{app}\left(T(C'), \texttt{app}(CX, \texttt{app}(CX, \texttt{app}(C, Q)))\right)$$

The quantum circuit calculus defines a rewriting rule to cancel two adjacent CX gates, which is introduced as the following precondition to the proof goal:

$$\mathbf{P_1} : \ \forall Q_1, \ \texttt{app}\left(CX, \texttt{app}(CX, Q_1)\right) \equiv Q_1$$

By linking with the specification of T in the CertiQ library, a new precondition about T can be also introduced as:

$$\mathbf{P_2} : \ \forall C_2 \ Q_2, \ \texttt{app}\left(T(C_2), Q_2\right) \equiv \texttt{app}\left(C_2, Q_2\right)$$

The CertiQ verifier then encodes the proof goals and preconditions into SMT formulas and invokes Z3 to check if the following formula is satisfiable:

$$\mathbf{P_1} \wedge \mathbf{P_2} \wedge \neg \mathbf{G}$$

If the above formula is satisfiable, the compiler pass is incorrect and a counter-example is generated by the verifier. Otherwise when the above formula is unsatisfiable, we successfully verify that the compiler pass written in CertiQ correctly preserve the semantics.

In the next sections, we will introduce all CertiQ components in detail. We will start with the most important, the quantum circuit calculus, that enables the efficient check of the equivalence of quantum circuits.

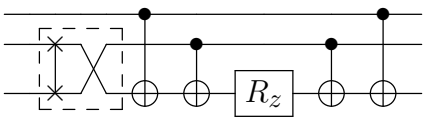
5.4 The Verified CertiQ Library

The CertiQ verified library follows the design of Qiskit to ease the learning curve for Qiskit programmers. This library implements a subset of Qiskit's functionality (without supporting pulses, schedules, etc.) that provides all infrastructure needed in writing new compiler passes, including Python classes for different representations of qubits , quantum gates, and quantum circuits, as well as a wide range of functions for circuit rewriting and representation conversion.

List-based circuit implementation. Unlike Qiskit's DAG implementation of quantum circuits that is heavily dependent on third party libraries, CertiQ implements the quantum circuit class based on generic Python lists. With appropriate assumptions that Python lists are correctly implemented, we can replace Qiskit's DAG implementation (which is hard to reason about) with CertiQ's list implementation and simplify the verification of compiler passes.

Verified circuit transformations. Based on the basic rewriting rules in Fig. 5.7, CertiQ provides a library of high-level verified circuit transformations that can be used to build more complicated compiler passes. For example, we implement and verify the `commutation_cancellation` transformation function in the IBM Qiskit compiler by composing the commutation rule and gate cancellation rule in Fig. 5.7.

Another example is the *routing* passes, which are often implemented in a way such that semantics preservation is broken in the middle of the pass and is restored later. For example, the semantics are not preserved right after inserting a $SWAP$ gate until the circuit layout is updated for the following gates. In CertiQ, we encapsulate the gate swapping and the layout updates in a single transformation function, which can be verified to preserve semantics before and after this transformation using the swap rule in Fig. 5.7. This transformation function is very useful to implement and verify compiler passes such as those meant for *routing*. The circuits in Fig. 5.3 provide an example. By invoking this transformation (shown in the following dashed box and implemented as the `swap_and_update_gate` method in the library) to the un-mapped circuit in Fig. 5.3,



we can generate the mapped circuit in Fig. 5.3 without exposing the intermediate inequivalence of quantum circuits to the level of symbolic executions. Thus, this verified transformation function enables the automated verification of *routing* passes.

Verified quantum data conversions. Converting quantum data between different representations enables powerful quantum circuit optimizations in the compiler. The verification of quantum compilers must also provide a correctness guarantee for these data conversions as they are much less intuitive and more error prone as compared to the classical case. The CertiQ compiler provides quantum data converters for basic quantum data structures, including conversions between the list-based circuit implementation and the Qiskit circuit implementation and conversions between the qubit IR representation and the Bloch sphere representation.

To verify the correctness of such conversions, we have to first define the *conversion relation* (denoted as $\sim$) between different data representations. This conversion relation does not have to be 1-to-1 correspondence. For example, we can say that a list implementation C_l and a DAG implementation C_d are convertable, i.e., $C_l \sim C_d$, if C_l and C_d have the same quantum registers and C_l is a topological sorting of C_d.

We can then use forward simulation [113] to show that the relation $\sim$ is valid and the conversion is sound. Two representations D_a and D_c of the same data D can be safely converted if and only if the simulation diagram holds as shown in Fig. 5.5. The simulation diagram says that, starting from two related states D_a and D_c, the resulting states D'_a and D'_c by applying any valid transformation t must still be related. Note that the transformation t may be implemented differently for different representations.

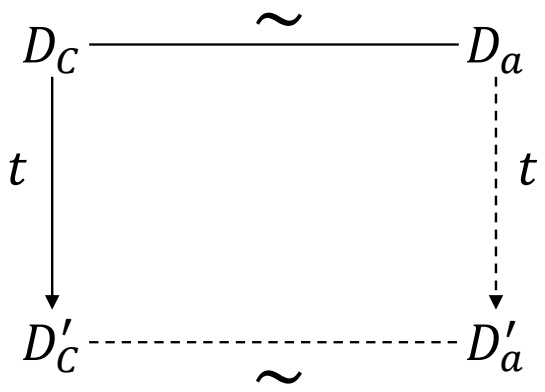


Figure 5.5: The simulation diagram defines the condition in which data representations can be converted safely.

Take the conversion between the Bloch sphere representation and the qubit state vector representation as an example. For single qubit optimizations, IRs of qubits (and 1-qubit gates) are often expanded to the state vector (matrix) representation. Further, it will be transformed to the Bloch sphere representation. Specifically, Bloch sphere representation (θ, ϕ)) is a projection of a general qubit state vector $|\psi\rangle = e^{i\gamma}\cos(\theta/2)\,|0\rangle + e^{i\gamma}e^{i\phi}\sin(\theta/2)\,|1\rangle$, where the global phase γ of a qubit state $|\psi\rangle$ is omitted. If we use this projection as the conversion relation, this relation can be written as the following Python code:

```
def Bloch_rep(gamma, theta, phi): return (theta, phi)
```

However, the simulation diagram in Fig. 5.5 does not hold for this relation when the transformation t is `tensor_product` or any other multi-qubit operations. The simulation diagram breaks because the untracked phase γ will induce a 2-qubit phase gate (Controlled Z rotation) between qubits beyond the 1-qubit case. This relative phase change will induce non-trivial quantum computation that is not revealed in the Bloch representation. To address this issue, in CertiQ, we explicitly exclude conversions between the Bloch sphere representation and other representation for cases beyond 1 qubit. CertiQ used this conclusion to detect a severe bug in Qiskit (see §5.8.1).

5.5 Quantum Circuit Calculus

It is intractable to check the equivalence of quantum circuits using the matrix representation (or denotational semantics) directly, due to the exponential computational cost. To address this challenge, we introduce symbolic representations and executions for quantum circuits, and rules to reduce gates and perform circuit rewriting.

Before diving into the calculus, we first define the syntax and semantics of quantum circuits supported by CertiQ. Our syntax is similar to OpenQASM [31] and can be transformed to and from the DAGCircuit representation in the Qiskit compiler.

Quantum circuit syntax. The syntax of quantum programs (or circuits) in CertiQ can be viewed as a subset of the OpenQASM standard (see Appendix §C.1). Features in OpenQASM that are

not supported by existing hardware such as classical control flow are not included in our syntax. Nevertheless, this syntax is general enough to support a wide range of gate representations, circuit transformations, and various targeting hardware. Detailed explanations of the syntax is omitted here and can be referred to OpenQASM standard [31].

$$\begin{aligned} [\![U]\!]_{\text{nqreg}} &:= \quad \texttt{matrix}(U_{q_1,\ldots,q_n}) \otimes I_{\texttt{qreg}\setminus\{q_1,\ldots,q_n\}} \\ [\![C_1\ ; C_2]\!]_{\text{nqreg}} &:= \quad [\![C_1]\!]_{\text{nqreg}} \times [\![C_2]\!]_{\text{nqreg}} \end{aligned}$$

Figure 5.6: Denotational semantics of quantum circuits and unitary operations in CertiQ. `matrix` denotes the unitary matrices of the quantum operations.

Denotational semantics. Figure 5.6 shows the denotational semantics of a quantum circuit C, which is defined as its corresponding unitary matrix and is denoted as $[\![C]\!]_{\text{nqreg}}$, where nqreg is the number of qubits in the quantum register used in the circuit. For example, the denotational semantics of an empty circuit with nqreg qubits is the identity matrix of size nqreg. The semantics for a quantum gate is the tensor product of its matrix representation on its qubit operands and identity matrix on other unrelated qubits. The semantics of the concatenation of two quantum programs is the multiplication of their matrix representations. The equivalence of two quantum circuits C_1 and C_2 is then defined using the equality of their denotational semantics:

$$\forall \text{nqreg},\ [\![C_1]\!]_{\text{nqreg}} = [\![C_2]\!]_{\text{nqreg}}$$

Symbolic representation and execution. A multi-qubit quantum register Q is symbolically represented as an array of symbolic qubits $(q_1, \cdots, q_n)$. CertiQ defines the symbolic function $\texttt{app}_{1q}(C, q)$ to denote the resulting qubit of applying C on q, and defines $\texttt{app}_{2q}(C, q_1, q_2, k)$ to denote the k-th resulting qubit ($k \in \{1, 2\}$) of applying the 2-qubit gate C on q_1 and q_2. The result for applying the whole circuit to a symbolic quantum register is represented as $\texttt{app}(C, Q)$, which

can be executed by applying each gate in sequence on Q:

$$\begin{aligned}
\texttt{app}(skip, Q) &\rightharpoonup Q \\
\texttt{app}(C_1; C_2, Q) &\rightharpoonup \texttt{app}(C_2, \texttt{app}(C_1, Q)) \\
\texttt{app}(U(i), (q_1, \ldots, q_n)) &\rightharpoonup (q_1, \ldots, \texttt{app}_{1q}(U, q_i), \ldots, q_n) \\
\texttt{app}(U(i,j), (q_1, \ldots, q_n)) &\rightharpoonup \\
&\quad (q_1, \ldots, \texttt{app}_{2q}(U, q_i, 1), \ldots, \texttt{app}_{2q}(U, q_j, 2), \ldots, q_n)
\end{aligned}$$

For example, applying the GHZ circuit

$$GHZ := H(0); CX(0,1); CX(1,2)$$

on the register (q_1, q_2, q_3) will result in (q_1', q_2', q_3') such that

$$\begin{aligned}
q_1' &= \texttt{app}_{2q}\left(CX, \texttt{app}_{1q}(H, q_1), q_2, 1\right) \\
q_2' &= \texttt{app}_{2q}\left(CX, \texttt{app}_{2q}(CX, \texttt{app}_{1q}(H, q_1), q_2, 2), q_3, 1\right) \\
q_3' &= \texttt{app}_{2q}\left(CX, \texttt{app}_{2q}(CX, \texttt{app}_{1q}(H, q_1), q_2, 2), q_3, 2\right)
\end{aligned}$$

Rewriting rules. To efficiently check the equivalence of symbolic representations of quantum circuits, CertiQ introduces a set of rules for reducing and rewriting gates that are inserted by compiler passes. For example, as shown in Fig. 5.7, the swap rules state that $SWAP$ gates will swap the two qubit arguments, which can be represented by the following lemmas:

$$\begin{aligned}
\texttt{app}_{2q}(SWAP, q_1, q_2, 1) &\equiv q_2 \\
\texttt{app}_{2q}(SWAP, q_1, q_2, 2) &\equiv q_1
\end{aligned}$$

The cancellation rules applying two adjacent CX gates on the same pair of qubits will not change the state:

$$\begin{aligned}
\texttt{app}_{2q}(CX, \texttt{app}_{2q}(CX, q_1, q_2, 1), \texttt{app}_{2q}(CX, q_1, q_2, 2), 1) &\equiv q_1 \\
\texttt{app}_{2q}(CX, \texttt{app}_{2q}(CX, q_1, q_2, 1), \texttt{app}_{2q}(CX, q_1, q_2, 2), 2) &\equiv q_2
\end{aligned}$$

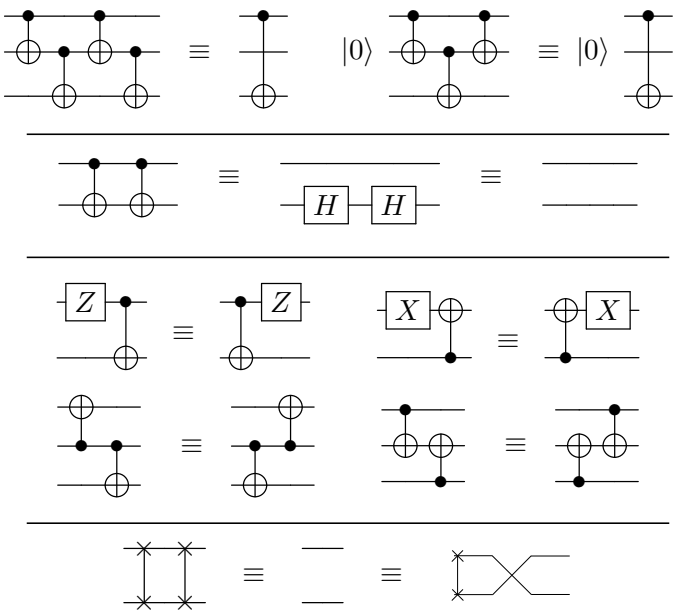


Figure 5.7: Examples of rules for reducing and rewriting gates. They are bridging rules (above), cancellation rules (2nd line), commutativity rules (3rd, 4th line), and swap rules (bottom).

These rewriting rules are defined as Z3 `assertions` about the symbolic functions $\texttt{app}_{1q}$ and $\texttt{app}_{2q}$, and are added into the list of preconditions of the proof obligations.

Lemmas for circuit composition. In the verification of compiler passes containing loops such as routing passes, we have to reason about sub-circuits or the first i gates of a circuit. CertiQ provides a set of shortcut lemmas to facilitate the compositional verification of quantum circuits (see Appendix C.2).

Soundness proofs. The symbolic execution, rewriting rules, and the shortcut lemmas are treated as axioms in CertiQ and are proven using the denotational semantics with the Coq proof assistant [30]. The soundness of the symbolic execution can be trivially proven by showing that output state is always the same denotational semantics with the input state. Since all rewriting rules only deal with a small number of gates, their soundness using matrix representation can also be proven easily. The shortcut lemmas are proven using the compositionality and linearity of unitary matrices.

Note that it is unnecessary to automate these soundness proofs in the quantum circuit calculus since they only need to be done once for all verification.

5.6 The CertiQ Synthesizer

The CertiQ synthesizer aims to provide language-level support for writing, specifying, and maintaining compiler passes. It consists of two components: a parser that parses the annotated user code into verification conditions and a wrapper library for user code to run in Qiskit.

5.6.1 Generating verification conditions

The CertiQ interface is a Python-like language (supporting branches, loops, and function calls) for writing compiler passes with annotations (see Appendix C.3). Because the SMT solver in the CertiQ verifier does not support branch statements and loops that are widely used in compiler passes, the synthesizer has to parse the annotated code into verification conditions (VCs) that are acceptable by the verifier.

Branch statements. The CertiQ synthesizer expands all branch statements. The verification condition of a branch is expanded into two separate VCs: one for each branch. The "true" branch condition will be added to the list of preconditions for a "true" branch implementation while negation of the branch condition will be added to the list of preconditions for the "false" branch. These preconditions are then used to generate further VCs. Note that CertiQ requires that all branch conditions must be representable as SMT formulas to enable the verification.

Although this expansion approach may lead to an exponential number of VCs, fortunately, the number of branches remains tractable in Qiskit compiler passes. For all 26 passes that we have verified, the number of branches is at most five.

Loop statements. Loops are unavoidable in compiler implementations, but SMT solvers cannot handle variable-length loops. Thus, we must provide additional information, such as loop

invariants. Fortunately, in this domain-specific environment (a quantum compiler), this problem can be easily and practically solved as loops in quantum compilers often follow fixed patterns. CertiQ provides *three* loop templates as library functions that take the loop body as an argument. We found these three loop templates can cover most of the Qiskit compiler passes. For example, `iterate_all_gates(circ, func)` is one template for loops that iterates over all the gates in `circ` and applies the same loop body `func` to each gate.

To synthesize such a loop template from the original implementation, CertiQ requires users to specify the variable name in the loop body that should be mapped to the `circ` argument of `iterate_all_gates(circ, func)`:

```
for gate in qcirc.gate_list:
    #@ circ: new_qcirc
    ... # loop body generates new_qcirc with gate
```

Loop templates provided by CertiQ pre-define loop invariants as pre- and postconditions about the loop body. In the below `iterate_all_gates` example, this loop template maintains an optimized version of a prefix of the original circuit while adding new gates one by one. The invariant should be that the circuit referred to as `new_circ` within the loop should be equivalent to the first i gates in the original circuit at i-th iteration of the loop. This loop template is written as follows:

```
def iterate_all_gates(circ, func):
    # subgoal
    assertion.push()
    i = Int("i")
    n = circ.size()
    cur_circ = QCircuit()
    assertion(i >= 0)
    assertion(i + 1 < n)
    assertion(equivalent_part(cur_circ, circ, i))
    new_circ = func(cur_circ, circ[i])
    certiq_prove(equivalent_part(new_circ, circ, i+1))
    assertion.pop()

    ret_dag = DAG()
    assertion(
        equivalent_part(ret_circ, circ, circ.size()))
    return ret_circ
```

We can see that `iterate_all_gates()` defines loop invariants as pre- and postconditions of

the loop body stored as `func`. In the verification process, when CertiQ invokes this loop template, it will generate and try to prove the subgoal that when the current `new_circ` (represented by `cur_circ` in the specification) is equivalent to the first i gates of `circ` before the i-th iteration, the `new_circ` after this iteration must be equivalent to the first $i + 1$ gates of `circ`. The specification states that under the preconditions, the result of the loop is a circuit that is equivalent to `circ`.

The other two loop templates provided by CertiQ are (1) `while_gate_remaining()` that is used in transformation passes such as `lookahead_swap`, and (2) `collect_runs()` that is used in passes such as `optimize_1q_gate` (§5.8.1). Together with `iterate_all_gates`, these three templates help CertiQ prove all loops in the 26 passes of Qiskit.

External function calls. For third party libraries and complicated functions using unsupported language features or data structures, CertiQ allows the programmer to label the function as an external function and specify its behavior using the "@external" decorator. The synthesizer will add the specification of an external function to the precondition list of programs invoking that external function. External functions will not be verified by CertiQ with respect to their specifications and, thus, are in our trusted computing base (TCB).

5.6.2 Generating execution code

In CertiQ, compiler passes rely on the verified CertiQ library, thus not directly compatible with the Qiskit framework. To generate compiler passes that can be integrated into Qiskit, CertiQ provides a Qiskit wrapper for CertiQ compiler passes that utilizes the data conversion library provided by the verified CertiQ library. The Qiskit wrapper performs the following steps to incorporate CertiQ passes into the Qiskit compilation process: 1) it converts the input DAG circuit from the Qiskit compilation flow to the OpenQASM IR; 2) then it invokes the CertiQ pass and receives the compiled circuit from the CertiQ pass; 3) it converts the compiled CertiQ circuit to the corresponding DAG circuit. With these steps, the Qiskit wrapper allows CertiQ integration into Qiskit.

5.7 The CertiQ Verifier

The CertiQ verifier generates proof obligations in the form of SMT formulas and invokes Z3 to automate the proof.

Generating proof obligations. In CertiQ, users do not need to explicitly provide specifications to the verifier as in other automated verification frameworks such as Yggdrasil [165]. Instead, CertiQ will load the pre-defined proof obligations of all seven types of compiler passes. For example, for all six types of compiler passes (excluding the analysis passes), we specify a proof obligation that the input and output circuits are equivalent through the `test()` method:

```
class TransformationPass():
    @classmethod
    def test(cls):
        optimizer = cls()
        init_circ = QCircuit()
        out_circ = optimizer.run(init_circ)
        certiq_prove(equivalent(out_circ, init_circ))
        print(cls.__name__ + " verified")
```

In the above `test()` method, CertiQ first generates symbolic circuits `init_circ` to represent the input quantum circuits, then symbolically executes the pass implementation through `optimizer.run`, and finally attempts to verify the proof obligations of circuit equivalence. When users implement a customized, device-independent transformation pass, they can use the `TransformationPass` virtual class as parent class to guide the generation of proof obligations:

```
class MyOwnPass(TransformationPass):
    #implementation omitted
    #...
```

Verification engine. Our verification engine is similar to that of Alive [111] and Yggdrasil [165] and maintains a *stack* of proof obligations. One difference is that CertiQ uses ArrayEx (Theory of Array) but Alive and Yggdrasil uses BV or QF_BV (quantified/quantifier-free bitvector) theories. A proof obligation is a precondition list paired with a claim list meaning that the list of claims should hold under the list of preconditions. At the beginning, the preconditions and the claims of a function will be pushed onto the stack as a proof obligation. The CertiQ verification engine

will keep popping the top proof obligation from the stack until it becomes empty. The claims of a function will automatically unfold through symbolic execution. When executing a specification, the engine will add the guarantees in the specification to the precondition list of the current proof obligation. If the specification contains preconditions, the verification engine will create a new proof obligation where the precondition list is the current precondition list, and the claim list holds this precondition of the specification. Such a new obligation will then be pushed into the stack. Once a proof obligation is fully unfolded, the verification engine will use Z3 to check whether the list of claims holds under the list of preconditions.

5.8 Case Studies

After introducing all of the technical details, we present two case studies to show how the CertiQ framework can detect quantum-specific bugs in the Qiskit compiler.

5.8.1 The `optimize_1q_gate` pass

We first focus on the verification of the `optimize_1q_gate` pass and show that, using CertiQ, we can reveal bugs that only arise in quantum software.

We verify the re-implemented `optimize_1q_gate` pass using the `merge_1q_gate` method. This pass collapses a chain of single-qubit gates into a single, more efficient gate [118], to mitigate noise accumulation. It operates on u_1, u_2, u_3 gates, which are native gates in the IBM devices. These gates can be naturally described as linear operations on the Bloch sphere, for example, u_1 gates are rotations with respect to the Z axis. For clarity, we list their matrix representations in Table 5.1.

The `optimize_1q_gate` pass has two function calls. First, it calls the `collect_runs` method to collect groups of consecutive u_1, u_2, u_3 gates. Then it calls `merge_1q_gate` to merge the gates in each group. `merge_1q_gate` (Fig. 5.8) first transforms the single qubit gates from the Bloch sphere representation to the unit quaternion representation [58], then the rotation merges

$$u_1(\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix}, \quad u_2(\phi, \lambda) = \frac{\sqrt{2}}{2} \begin{pmatrix} 1 & -e^{i\lambda} \\ e^{i\phi} & e^{i(\lambda+\phi)} \end{pmatrix}$$

$$u_3(\theta, \phi, \lambda) = \frac{\sqrt{2}}{2} \begin{pmatrix} \cos(\theta) & -e^{i\lambda}\sin(\theta) \\ e^{i\phi}\sin(\theta) & e^{i(\lambda+\phi)}\cos(\theta) \end{pmatrix}$$

Table 5.1: Matrix representation of physical gates u_1, u_2 and u_3. u_1 is a Z rotation on the Bloch sphere.

are performed in that representation.

As described in Section 5.4, conversion between these two representations allow only single-qubit operations to adhere to the simulation diagram. However, in Qiskit, all gates can be modified with a `c_if` or `q_if` method to condition its execution on the state of other classical or quantum bits. When the transpiler pass attempts to optimize these conditional gates, it can lead to an incorrect circuit. For this reason, in the implementation of this pass, we have to include that `gate1.q_if == False` and `gate1.c_if == False` unless both are controlled and the control bits are the same.

The bug described above, which relates to how quantum circuit instructions can be conditioned, have been observed in Qiskit in the past [146, 181]. In the absence of rigorous verification like in this work, such bugs are hard to discover. In practice, this is usually done via extensive randomized testing of input and output circuits, which does not provide any guarantee of finding faulty code. The results here demonstrate that our *forward simulation* technique for `merge_1q_gate` is effective for detecting quantum-related bugs.

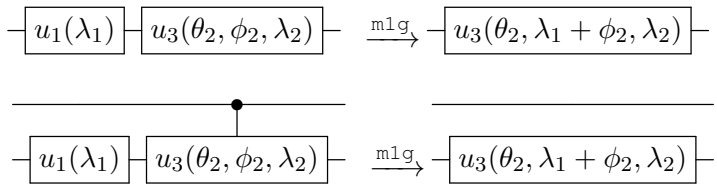
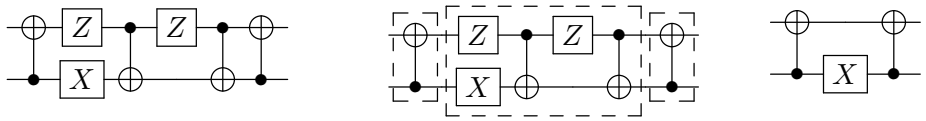


Figure 5.8: Correct execution (top) and incorrect execution (bottom) of `merge_1q_gate`.

5.8.2 `commutation` passes

`commutation_analysis` and `commutative_cancellation` are a pair of compiler passes that optimize Qiskit DAGCircuits using the quantum commutation rules and the cancellation rules pictured in Fig. 5.7. First, `commutation_analysis` transforms the quantum circuit to a representation called commutation groups [160] where nearby gates that commute with each other are grouped together. Next, `commutative_cancellation` performs cancellation inside the newly formed groups. In Fig. 5.9, we give a working example.



(a) (b) (c)

Figure 5.9: A working example of `commutation_analysis` and `commutative_cancellation`. (a) The un-optimized circuit, (b) `commutation_analysis` forming the commutation groups, and (c) `commutative_cancellation` cancels self-inverse gates inside groups.

We find two bugs when re-implementing these two passes. First, the commutation group can be viewed as another representation of the quantum circuit. However, when we try to convert the symbolic quantum circuit representation to the commutation group representation defined in the `commutation_analysis`, we find that the commutation group representation does not satisfy the simulation diagram defined in Section 5.4. We found this violation comes from the fact that the commutation relation is in general not transitive. For example, if we denote the commutation relation as $\sim$ and there is 3 quantum gates, A, B, C where $A \sim B$, $B \sim C$, then $A \sim C$ is not guaranteed to be true. For this reason, gates with pairwise commutation relations cannot be grouped together. We propose two solutions to this bug. First, we can make sure the circuits that these passes operate on have a limited gate set where $\sim$ is indeed transitive. For example, in the gate set {CX, X, Z, H, T, u_1, u_2, u_3}, $\sim$ is transitive. Second, we can use a new algorithm that does not assume transitivity.

5.8.3 `routing` passes

For routing passes, there are three proof obligations for `swap` passes:

- The pass must be semantics preserving.
- The output DAGCircuit of the pass must conform to the coupling map of the physical device.
- The pass must terminate.

The first proof obligation is same with all other passes. For the second proof obligation that ensures the pass correctly accomplishes its goal of transforming the circuit to match device coupling constraints, we must verify that every 2-qubit gate in the output circuit operates on two neighboring physical qubits. For the third proof obligation, CertiQ does not attempt to find a complete solution because it is undecidable. Instead, CertiQ aims to provide sound termination analysis for practical implementations. First, CertiQ concretizes the problem to verify the termination of passes with an input circuit of bounded depth on a given coupling map by switching to the bitvector model in the Z3 SMT solver. Termination can be proved by constructing strictly monotonic functions in a finite domain. For program states that are not in a loop or a recursive function, the program counter is a monotonic function to provide a termination guarantee. For variable-length loops and while loops, CertiQ allows users to provide a monotonic function, for example,

```
gates_remaining = circ.gate_list()
while gates_remaining != 0:
# @mono: -gates_remaining.size
  ... # implementation code
```

Then in the backend SMT solver in the verifier solves for the circuit input that keeps `gates_remaining.size` unchanged and gives it as a counter example.

We verified two routing passes: `basic_swap`, `lookahead_swap`. We report that all two passes comply with the first two proof obligations. However, we find a counter-example circuit on coupling map of the IBM 16 qubit device, where the `lookahead_swap` pass does not terminate on (see Fig. 5.10). The `lookahead_swap` pass greedily finds the next best 4 `swap` gates to minimize the total distance of the unmapped 2-qubit gates. However, the counter example we

found shows that the 4 `swap` gates can cancel through applying the swap rules in Fig. 5.7 and `gate_remaining.size` will not update. This bug can be fixed by introducing randomization to break ties when needed.

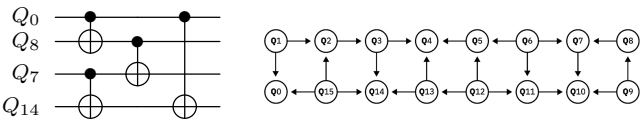


Figure 5.10: (left) A counter-example generated by CertiQ that shows Qiskit's `lookahead_swap` pass does not always terminate on the IBM 16 qubit device. (right) The coupling map of the IBM 16 qubit device. Arrows indicate available CNOT directions (which does not affect the swap insertion step).

5.9 Evaluation

In CertiQ, we implemented 26 out of 30 compiler passes from Qiskit. For several passes, such as routing passes, we wrote two versions, one calling the verified CertiQ transformation library to rewrite circuits, the other directly modifying the input circuits. All implementations of the 26 passes can be verified successfully and all the generated Qiskit passes passed the regression tests provided by Qiskit. The 30 passes are listed in Qiskit v0.192 website (the number becomes 42 passes in a recent update of Qiskit v0.200 during the submission). Among all 30 passes, 4 passes that we do not verify are: *StochasticSwap*, *FixedPoint*, *DAGFixed Point*, *CrosstalkAdaptiveSchedule*, which either include data structures we cannot support or are meta-optimizations. For example, *FixedPoint* is a compiler pass that controls the execution of another compiler pass and repeatedly executes that pass until two consecutive compilation gives the same result (thus, reaches a "fixed point"). We evaluated CertiQ based on the implementions of the 26 verified passes.

Automation level. CertiQ is mostly automated: when writing CertiQ compiler passes, the only additional verification effort is to write annotations for generating loop invariants (not the loop invariants themselves), as well as for external functions. Programmers will have to learn the CertiQ interface, which, however, is a limited burden to programmers who are already familiar with Qiskit.

Verification performance. For all compiler passes written with high-level transformation functions in the CertiQ library, their verification can finish within seconds. For passes that only apply low-level rewriting rules, their verification can be much slower than the ones using using high-level transformation functions but can still finish within one minute. The three benchmarks used are near-term quantum applications of 10, 20, 500 qubits, respectively. The results show that CertiQ does not compromise compilation performance. In theory, the CertiQ_lib function makes $O(n^2)$ circuit moves instead of $O(n)$ in the other implementations since every time it insert a swap, it will update all remaining gates in the circuit. However, we do not observe a significant performance loss because most of the computational time is spent on finding the optimized swap gates to insert.

5.10 Related Work

Quantum programming environments with a verifier. Several quantum programming environments support the verification of quantum programs running on them. For example, in the QWire quantum language [148, 147], programmers can use the embedded verifier based on the Coq proof assistant [30] to create mechanized proofs for their programs. The $Q\,|SI\rangle$ programming environment [109] allows users to reason about their programs written in the quantum **while**-language with quantum Floyd-Hoare logic [193]. In contrast to CertiQ, these environments require expertise both in quantum computing and mechanized verification to manually write proofs in proof assistants. Besides, these verifiers are built for verifying quantum programs rather than compiler passes to transform quantum programs.

Verified quantum-related compilers. Previous studies on compiler verification for reversible circuits [10], ancillae uncomputation [147] and compiler optimizations [78] utilize interactive theorem provers such as F* [119] and Coq [30] to conduct manual proofs, which do not provide an extensible interface for developers to verify future extensions with a low proof burden. In contrast, the CertiQ verification framework allows developers to write new compiler passes and automate their verification.

Algorithms to perform efficient quantum circuit equivalence checking have been discussed from the view of quantum algorithms [189], quantum communication protocols [18], and verification of compilation [9]. However, while powerful, these checking algorithms are too complicated for use in automated manner like what is possible with CertiQ verification.

Push-button verification in classical computing. Push-button verification has been applied to building verified compilers, file systems, OS kernels, and system monitors [165, 130, 129]. Our technical choice and many of the verification ideas are heavily influenced by these previous works. However, these systems cannot support variable-length loops, calculus of quantum circuit equivalence, and our contract-based reasoning.

CHAPTER 6

CONCLUSION

6.1 Thesis Review

We are in an era that one might, for the first time, be able to witness the realization of practical quantum computers. Building a quantum computer that surpasses the computational power of its classical counterpart is a great engineering challenge. Quantum software optimizations and hardware-software co-design can provide an accelerated pathway to the first generation of quantum computing applications that might save years of engineering effort. Current quantum software stacks follow a layered approach similar to the stack of classical computers, which was designed to manage the complexity. In this thesis, we point out that greater efficiency of quantum computing systems can be achieved by breaking the abstractions between these layers.

In Chapter 3, we developed a method to combine ideas from classical scheduling algorithms and quantum control theory to improve circuit latency of near-term quantum applications. This compilation aggregates multi-qubit instructions and in this way breaks the ISA abstraction in the standard gate-based compilation scheme, resulting in a competitive pulse time reduction. Our implementation of this compilation methodology shows that in several important near-term quantum applications, our compilation process achieves up to 10X circuit latency reduction on superconducting architectures, which helps enable many appealing applications. We further analyze how different program characteristics, including parallelism, commutativity, and connectivity, interact with the level of optimization by instruction aggregation. We observe that our compilation scheme is most advantageous for quantum circuits that are highly serial, have low spatial locality, and utilize sophisticated information encoding.

In Chapter 4, we discussed the problem of fault tolerantly preparing approximate Gottesman-Kitaev-Preskill states. GKP states appear to be amongst the leading candidates for correcting errors when encoding qubits into oscillators. However the preparation of GKP states remains a significant theoretical and experimental challenge. Until now, no clear definitions for fault-tolerantly

preparing GKP states have been provided. Without careful consideration, a small number of faults can lead to large uncorrectable shift errors. After proposing a metric to compare approximate GKP states, we provide rigorous definitions of fault-tolerance and introduce a fault-tolerant phase estimation protocol for preparing such states. The fault-tolerant protocol uses one flag qubit and accepts only a subset of states in order to prevent measurement readout errors from causing large shift errors. We then show how the protocol can be implemented using circuit QED. In doing so, we derive analytic expressions which describe the leading order effects of the non-linear dispersive shift and Kerr non-linearity. Using these expressions, we show that to mitigate the non-linear dispersive shift and Kerr terms would require the protocol to be implemented on time scales four orders of magnitude longer than the time scales relevant to the protocol for physically motivated parameters. Despite these restrictions, we numerically show that a subset of the accepted states of the fault-tolerant phase estimation protocol maintain good error correcting capabilities even in the presence of noise.

In Chapter 5, we presented CertiQ, a verification framework for writing and verifying compiler passes of Qiskit, the most widely-used quantum compiler. To our knowledge, CertiQ is the first effort enabling the verification of real-world quantum compiler passes in a mostly-automated manner. Compiler passes written in the CertiQ interface with annotations can be used to generate verification conditions, as well as the executable code that can be integrated into Qiskit. CertiQ introduces the quantum circuit calculus to enable the efficient checking of equivalence of quantum circuits by encoding such a checking procedure into an SMT problem. CertiQ also provides a verified library of widely-used data structures, transformation functions for circuits, and conversion functions for different quantum data representations. This verified library not only enables modular verification but also sheds light on future quantum compiler design. We have re-implemented and verified 26 (out of 30) Qiskit compiler passes in CertiQ, during which three bugs are detected in the Qiskit implementation. Our verified compiler pass implementations passed all of Qiskit's regression tests without showing noticeable performance loss.

6.2 Future Directions for Near-term Quantum Architecture Design

In this thesis, we proposed that greater quantum efficiency can be achieved by breaking abstraction layers in the quantum computing stack. In Chapter 3 and Chapter 4, we examined two works that are closing the gap between current quantum technology and real-world quantum computing applications. We would also like to briefly discuss some promising future directions along this line.

6.2.1 Pulse-level compilation

To maximize the potential of the current pulse-level compilation framework, there are several future directions that are worth exploring:

- In the future, it could be interesting to explore micro-controller designs that provide hardware-efficient gradient-based optimizations and fast feedback control, which could leverage current technologies in accelerators for Deep Learning. This attempt might lead to unexpected new designs of classical control system for QC platforms.

- It's a promising and important direction to develop new pulse optimizations that are algorithm-aware. GRAPE-based pulse generation in the work described in Chapter 3 performs optimizations in a high-dimensional parameter space and can become ineffective due to the high dimensionality. With the information from the specific quantum applications being optimized, the number of parameters can be largely reduced. The simplest example is the "partial pulse engineering" technique. In a quantum chemistry application, if we know an aggregated instruction is a Jordan-Wigner string with only the rotation angle of one qubit is varied, we can fix the pulse control fields of all Hamiltonian terms but the one that controls the rotation angle of that qubit. In this case, the optimization of this instruction becomes a 1-dimensional problem and doesn't scale with the instruction size.

- It will also be interesting to look into methods to achieve pulse-level parallelism for efficient

scheduling. With my current work being the first compilation scheme to adapt commutation relations for gate-level parallelism, it's worthwhile to look into a more fine-grained pulse-level parallelism. For example, by utilizing the Baker-Campbell-Hausdorff formula in quantum mechanics to cancel the extra unitaries generated by altering the time order, two non-commuting aggregated instructions can be parallelized with high-precision approximation.

6.2.2 Noise-Tailoring Compilation

We can further explore the idea of breaking the ISA abstraction. Near-term quantum devices have errors from elementary operations like 1- and 2-qubit gates, but also emergent error modes like cross-talk. Emergent error modes are hard to characterize and to mitigate. Recently, it has been shown that randomized compiling could transform complicated noise channels including cross-talk, SPAM errors and readout errors into simple stochastic Pauli errors [191], which could potentially enable subsequent noise-adaptive compilation optimizations. We believe if compilation schemes that combine noise tailoring and noise adaptation could be designed, they will outperform existing compilation methods.

6.2.3 Algorithm-Level Error Correction

Near-term quantum algorithms such as Vairiational Quantum Eigensolver (VQE) and Quantum Approximate Optimization Algorithm (QAOA) are tailored for NISQ hardware, breaking the circuit/ISA abstraction. We could take a step further and look at high level algorithms equipped with customized error correction/mitigation schemes. Prominent examples of this idea are the Generalized Superfast Encoding (GSE) [158] and the Majorana Loop Stabilizer Code (MLSC) [83] for quantum chemistry. In GSE and MLSC, the overhead of mapping Fermionic operators onto qubit operators stays constant with the qubit number (as opposed to linear scaling in the usual Jordan-Wigner encoding or logarithmic in Bravyi-Kitaev encoding). On the other hand, qubit operators in these mappings are logical operators of a distance 3 stabilizer error correction code so that we can

correct all weight 1 qubit errors in the algorithm with stabilizer measurements. These work are the first attempts to algorithm-level error correction and we are expecting to see more efforts of this kind to improve the robustness of near-term algorithms.

6.2.4 Dissipation-Assisted Error Mitigation

We generally think of dissipation as competing with quantum coherence. However, with careful design of the quantum system, dissipation can be engineered and used for improving the stability of the underlying qubit state. Previous work on autonomous qubit stabilization [112] and error correction [90] suggest that properly engineered dissipation could largely extend qubit coherence time. Exploring the design space of such systems and their associated error correction/mitigation schemes might provide alternative paths to an efficient and scalable quantum computing stack.

6.3 Future Directions for Verification of Quantum Computation

On the verification side, we discussed verifying quantum compilation in a realistic environment using model checking. However, there are still more aspects of quantum computing that formal verification could be used for robust and reliable development and implementation.

6.3.1 Verified Quantum Toolchain

The ultimate goal of combining formal verification and quantum computing is to build a formally verified QC system. The CertiQ verification work in Chapter 5 focused on the compilation stage, we expect to extend the developed methods in CertiQ to high-level quantum algorithm libraries, hardware description language of the classical control system and eventually the pulse-level implementation of the quantum computation. A similar approach for classical computing is already outlined in the current work of verified OS kernels, file systems and virtual machines. The difficulty in the quantum case will be the contradiction between heavy optimization across abstraction layers and the requirement of modular verification for maintaining complexity. This will require

novel design of certified abstraction layers in the verification framework.

6.3.2 Verified Fault Tolerant Quantum Computing

In classical computing, the verification of fault tolerance and error correction is not a pressing need as classical hardware are reliable. However, in fault tolerant quantum computing, error correction protocols are the fundamental building blocks of quantum machines with millions of qubits and most computing time will be devoted to execute error correction. We can think of error correction protocols as "quantum CPUs" with similar (but slightly smaller) scale of classical CPUs. Thus, to develop and implement such complicated systems, we expect that their correctness can only be guaranteed by formal verification. It will be interesting to investigate into methods to apply mechanized proofs to realistic fault tolerance protocols. For example, some implementations of Color code decoding algorithms involve more than 20,000 lines of C code. To prove the correctness of these decoding methods will require enormous man$\times$hour and only be possible by designing and applying highly-automated verification tactics in proof assistants.

On a high level, a formal verification framework can provide the QEC community with a standardized tool to compare, validate and improve QEC protocols, serving a role similar to SMT-LIB for SMT solvers with stronger verification support.

Appendices

APPENDIX A

PHYSICAL QUANTUM GATES

Below, we list some physical gates in different architectures:

- In platforms with Heisenberg interaction Hamiltonian, such as quantum dots [89], the directly directly implementable 2-qubit physical gate is the $\sqrt{\text{SWAP}}$ gate (which implements a SWAP when applied twice).

- In platforms with ZZ interaction Hamiltonian, such as superconducting systems of Josephson flux qubits [140, 139] and NMR quantum systems [188], the physical gate is the CPhase gate, which is identical to the CNOT gate up to single qubit rotations.

- In platforms with XY interaction Hamiltonian, such as capacitively coupled Josephson charge qubits ($e.g$ transmon qubits [102]), the 2-qubit physical gate is iSWAP gate.

- For trapped ion platforms with dipole-chain interaction, two popular physical 2-qubit gates are the geometric phase gate [106] and the XX gate [38].

APPENDIX B

NUMERIC SIMULATION DETAILS IN THE PHASE ESTIMATION PROTOCOL

B.1 Shift error arising from amplitude damping before the controlled-displacement gate

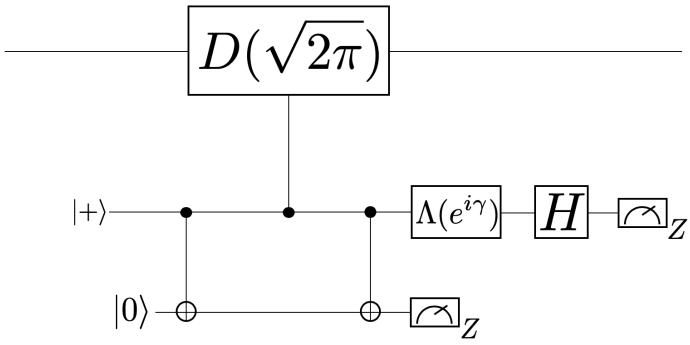


Figure B.1: Controlled-displacement circuit with the flag qubit. The protocol is aborted if the flag qubit measurement is non-trivial.

Consider amplitude damping on the $|+\rangle$ ancilla state before the controlled displacement gate as shown in Fig. B.1. Let $|\varphi\rangle$ be the input state part of the cavity Hilbert space. Defining p as the damping rate, after the controlled-displacement gate, the state of the system can be written as

$$|\psi^{(1)}\rangle = \frac{1}{\sqrt{2}}\Big\{ |00\rangle\,|\varphi\rangle + \sqrt{1-p}\,|11\rangle\,D(\sqrt{2\pi})\,|\varphi\rangle + \sqrt{p}\,|01\rangle\,|\varphi\rangle \Big\}. \tag{B.1}$$

Here we have omitted writing the state of the environment interacting with the first ancilla qubit. However, this does not affect the result of the calculations that follow.

After the second CNOT gate, the state becomes

$$|\psi^{(2)}\rangle = \frac{1}{\sqrt{2}}\Big\{ |00\rangle\,|\varphi\rangle + \sqrt{1-p}\,|10\rangle\,D(\sqrt{2\pi})\,|\varphi\rangle + \sqrt{p}\,|01\rangle\,|\varphi\rangle \Big\}. \tag{B.2}$$

If the flag qubit is measured as 1, the protocol is aborted. So assume that the flag is measured

as 0. In this case the state becomes (tracing out the flag qubit)

$$|\psi^{(3)}\rangle = \frac{1}{\sqrt{2-p}}\Big\{ |0\rangle\,|\varphi\rangle + \sqrt{1-p}\,|1\rangle\,D(\sqrt{2\pi})\,|\varphi\rangle \Big\}. \tag{B.3}$$

Applying the remaining gates in Fig. B.1, the final state prior to the measurement of the ancilla qubit is

$$|\psi^{(4)}\rangle = \frac{1}{\sqrt{2(2-p)}}\Big\{ |0\rangle\,(1+e^{i\gamma}\sqrt{1-p}D(\sqrt{2\pi}))\,|\varphi\rangle + |1\rangle\,(1-e^{i\gamma}\sqrt{1-p}D(\sqrt{2\pi}))\,|\varphi\rangle \Big\}. \tag{B.4}$$

If the measurement of the ancilla is 0, the output state will be

$$|\psi^{\text{out,0}}\rangle = \frac{1}{\sqrt{2-p}}\Big\{(1+e^{i\gamma}\sqrt{1-p}D(\sqrt{2\pi}))\,|\varphi\rangle \Big\}. \tag{B.5}$$

If the measurement of the ancilla is 1, the output state will be

$$|\psi^{\text{out,1}}\rangle = \frac{1}{\sqrt{2-p}}\Big\{(1-e^{i\gamma}\sqrt{1-p}D(\sqrt{2\pi}))\,|\varphi\rangle \Big\}. \tag{B.6}$$

Both outcomes occur with probability $1/2$.

B.2 Analytic derivation of the unitary operator describing the evolution of the qubit-cavity system during the implementation of the controlled-$D(\sqrt{2\pi})$ gate.

B.2.1 Implementation of the controlled-$D(\sqrt{2\pi})$ gate in the lab frame.

In this section, we will derive the unitary operator describing the evolution of the qubit-cavity state during the application of the control-displacement gate. In the dispersive regime, the qubit-cavity interaction can be described as:

$$H(t) = H_s + H_d(t), \tag{B.7}$$

where

$$H_s = \tilde{\omega}_r a^\dagger a + \tilde{\omega}_a Z + \chi a^\dagger a Z - \phi (a^\dagger a)^2 Z. \tag{B.8}$$

and

$$H_d(t) = \mathcal{E}(t)(a + a^\dagger). \tag{B.9}$$

In Eq. (B.8), $\phi = \frac{g^4}{\Delta^3}$, $\tilde{\omega}_r = \omega_r + \phi$, $\chi = \frac{g^2}{\Delta} - \phi$ and $\tilde{\omega}_a = \frac{\omega_a + \chi}{2}$. The term $\phi(a^\dagger a)^2 Z$ corresponds to the non-linear dispersive shift. Note that we have not included the Kerr term $-\frac{K}{2}(a^\dagger a)^2$. At the end of this section we will modify our results to take into account its effect. We also note that in writing the drive Hamiltonian in Eq. (B.9), we neglected a term of the form λX (where $\lambda = (g/\Delta)$ and all higher powers in λ. For parameter values considered in this paper, we found the effect of this term to be negligible. Additionally, we chose a pulse shape which is real valued.

In Eq. (B.9), we represent the drive pulse $\mathcal{E}(t)$ as

$$\mathcal{E}(t) = \Omega_x(t) \cos(\omega_d t) + \Omega_y(t) \sin(\omega_d t), \tag{B.10}$$

where $\omega_d = \tilde{\omega}_r - \chi$ is the drive frequency.

Since the Hamiltonian does not commute at different times, the unitary operator describing the time evolution under the Hamiltonian of Eq. (B.7) is given by

$$V(0, T) = \mathcal{T} e^{-i \int_0^T H(t') dt'}. \tag{B.11}$$

The right-hand side of Eq. (B.11) can be computed using the Suzuki-Trotter decomposition, so that

$$
\begin{aligned}
V(0,T) &= \Big(\lim_{n\to\infty} \prod_{j=1}^{n} e^{\frac{-iT}{n}(\tilde{\omega}_r + (\chi - \phi a^\dagger a)Z) a^\dagger a} e^{\frac{-iT}{n}\mathcal{E}(t_j)(a+a^\dagger)} \Big) e^{-iT\tilde{\omega}_a Z} \\
&= \Big(\lim_{n\to\infty} \prod_{j=1}^{n} e^{-i\tilde{t}(\omega_+ - \phi a^\dagger a) a^\dagger a} e^{-i\tilde{t}\mathcal{E}(t_j)(a+a^\dagger)} \Big) e^{-i\tilde{\omega}_a T} \left|0\right\rangle \left\langle 0\right| + \qquad \text{(B.12)} \\
&\Big(\lim_{n\to\infty} \prod_{j=1}^{n} e^{-i\tilde{t}(\omega_- + \phi a^\dagger a) a^\dagger a} e^{-i\tilde{t}\mathcal{E}(t_j)(a+a^\dagger)} \Big) e^{i\tilde{\omega}_a T} \left|1\right\rangle \left\langle 1\right|, \qquad \text{(B.13)}
\end{aligned}
$$

where $\tilde{t} \equiv T/n$, $\omega_\pm \equiv \tilde{\omega_r} \pm \chi$.

Now, the two terms on the right-hand side of Eq. (B.13) can be decomposed as

$$
\begin{aligned}
&\lim_{n\to\infty} \prod_{j=1}^{n} e^{-i\tilde{t}(\omega_+ - \phi a^\dagger a) a^\dagger a} e^{-i\tilde{t}\mathcal{E}(t_j)(a+a^\dagger)} \\
&= \lim_{n\to\infty} R_n^n (R_n^{-n} D_{t_n} R_n^n) \cdots (R_n^{-2} D_{t_2} R_n^2)(R_n^{-1} D_{t_1} R_n^1), \qquad \text{(B.14)}
\end{aligned}
$$

where

$$
R_n = e^{-i\tilde{t}(\omega_+ - \phi a^\dagger a) a^\dagger a}, \qquad \text{(B.15)}
$$

and

$$
D_{t_j} = e^{-i\tilde{t}\mathcal{E}(t_j)(a+a^\dagger)}. \qquad \text{(B.16)}
$$

Hence from the above, we see that we need to compute terms of the form

$$
R_n^{-k} D_{t_k} R_n^k = e^{i\tilde{t}k(\omega_+ - \phi a^\dagger a) a^\dagger a} e^{-i\tilde{t}\mathcal{E}(t_k)(a+a^\dagger)} e^{-i\tilde{t}k(\omega_+ - \phi a^\dagger a) a^\dagger a}. \qquad \text{(B.17)}
$$

In order to compute the products appearing in Eq. (B.17), we will use the identity $Ae^{B}A^{-1} = e^{ABA^{-1}}$. Defining,

$$H' = k(\omega_+ - \phi a^\dagger a)a^\dagger a, \tag{B.18}$$

and

$$A_{\tilde{t}} = e^{-i\tilde{t}\mathcal{E}(t_k)(a+a^\dagger)}, \tag{B.19}$$

with

$$A(\tilde{t}) = e^{iH'\tilde{t}}A_{\tilde{t}}e^{-iH'\tilde{t}} \tag{B.20}$$

we have that

$$A(\tilde{t}) = \exp\Big(- i\tilde{t}\mathcal{E}(t_k)e^{iH'\tilde{t}}(a + a^\dagger)e^{-iH'\tilde{t}}\Big), \tag{B.21}$$

The term $e^{iH'\tilde{t}}(a + a^\dagger)e^{-iH'\tilde{t}}$ in Eq. (B.21) can be computed using Heisenberg's equation of motion. We obtain

$$\frac{da(\tilde{t})}{d\tilde{t}} = i[H', a(\tilde{t})], \tag{B.22}$$

with

$$[H', a] = k(\phi' + 2\phi a^\dagger a)a, \tag{B.23}$$

where we defined

$$\phi' \equiv \phi - \omega_+. \tag{B.24}$$

We thus have the following set of coupled differential equations:

$$\frac{da(\tilde{t})}{d\tilde{t}} = ik(\phi' + 2\phi a^\dagger(\tilde{t})a(\tilde{t}))a(\tilde{t}), \tag{B.25}$$

and

$$\frac{da^\dagger(\tilde{t})}{d\tilde{t}} = -ika^\dagger(\tilde{t})(\phi' + 2\phi a^\dagger(\tilde{t})a(\tilde{t})), \tag{B.26}$$

The solutions to Eqs. (B.25) and (B.26) are given by

$$a(\tilde{t}) = ae^{ik\tilde{t}(\phi' + 2\phi(a^\dagger a - 1))}, \tag{B.27}$$

and

$$a^\dagger(\tilde{t}) = e^{-ik\tilde{t}(\phi' + 2\phi(a^\dagger a - 1))}a^\dagger. \tag{B.28}$$

To see this, we take a derivative of Eq. (B.27) to obtain

$$\frac{da(\tilde{t})}{d\tilde{t}} = ik\phi' a(\tilde{t}) + 2ik\phi a(\tilde{t})(a^\dagger a - 1). \tag{B.29}$$

Comparing Eqs. (B.25) and (B.29), we must have that

$$a(\tilde{t})(a^\dagger a - 1) = a^\dagger(\tilde{t})a^2(\tilde{t}). \tag{B.30}$$

Using Eqs. (B.27) and (B.28) and defining $H_{te} \equiv -kt(\phi' + 2\phi(a^\dagger a - 1))$, we have that

$$\begin{aligned} a^\dagger(\tilde{t})a^2(\tilde{t}) &= e^{iH_{te}\tilde{t}}a^\dagger e^{-iH_{te}\tilde{t}}e^{iH_{te}\tilde{t}}ae^{-iH_{te}\tilde{t}}ae^{-iH_{te}\tilde{t}} \\ &= a^\dagger e^{-2i\phi k\tilde{t}}ae^{2i\phi k\tilde{t}}ae^{-iH_{te}\tilde{t}} \\ &= a^\dagger aae^{-iH_{te}\tilde{t}} \\ &= a^\dagger aa(\tilde{t}), \end{aligned} \tag{B.31}$$

as desired. A similar calculation can be done for $a^\dagger(\tilde{t})$. Hence we have that

$$A(\tilde{t}) = \exp\Big(- i\tilde{t}\mathcal{E}(t_k)(ae^{i\Phi_k\tilde{t}} + e^{-i\Phi_k\tilde{t}}a^\dagger)\Big), \tag{B.32}$$

for

$$\Phi_k = k(\phi' + 2\phi(a^\dagger a - 1)). \tag{B.33}$$

Writing $V(0,T)$ (Eq. (B.13)) as

$$V(0,T) = V_+(0,T)\left|0\right\rangle\left\langle 0\right| + V_-(0,T)\left|1\right\rangle\left\langle 1\right|, \tag{B.34}$$

using Eq. (B.32) and the fact that $R_n^n = e^{-iT(\omega_+ - \phi a^\dagger a)a^\dagger a} \equiv R_+(T)$, we can write

$$V_+(0,T) = R_+(T)\lim_{n\to\infty}\prod_{k=1}^{n}\exp\Big\{A_k a^\dagger - aA_k^\dagger\Big\}, \tag{B.35}$$

where we defined

$$A_k \equiv -\frac{iT}{n}\mathcal{E}(t_k)e^{\frac{-iT}{n}\Phi_k}. \tag{B.36}$$

Notice that we can write $V_+(0,T)$ as products of displacements $D(A_k)$. However, now A_k is an operator instead of a complex number. In order to compute the products in Eq. (B.35), we will

need to obtain a relation for terms of the form $D(A_k)D(A_j)$ (where A_k is given in Eq. (B.36)). Since A_k is proportional to T/n and remembering that we will take the limit where $n \to \infty$, using Baker-Campbell-Hausdorff, we have the exact relation

$$\exp\Big\{A_k a^\dagger - a A_k^\dagger + A_j a^\dagger - a A_j\Big\} = D(A_k)D(A_j)\exp\Big\{-\frac{1}{2}[A_k a^\dagger - a A_k^\dagger, A_j a^\dagger - a A_j]\Big\} \tag{B.37}$$

The commutator on the right hand side of Eq. (B.37) has four terms

$$[A_k a^\dagger - a A_k^\dagger, A_j a^\dagger - a A_j] = [A_k a^\dagger, A_j a^\dagger] - [A_k a^\dagger, a A_j^\dagger] - [a A_k^\dagger, A_j a^\dagger] + [a A_k^\dagger, a A_j^\dagger]. \tag{B.38}$$

As we will show, two of these terms vanish when taking the limit $n \to \infty$.

First, we compute

$$[A_k a^\dagger, A_j a^\dagger] = \Big(\frac{-iT}{n}\Big)^2 \mathcal{E}(t_k)\mathcal{E}(t_j)[e^{-\frac{iT}{n}\Phi_k} a^\dagger, e^{-\frac{iT}{n}\Phi_j} a^\dagger], \tag{B.39}$$

with

$$\begin{aligned}[e^{-\frac{iT}{n}\Phi_k} a^\dagger, e^{-\frac{iT}{n}\Phi_j} a^\dagger] &= e^{-\frac{iT}{n}\Phi_k} a^\dagger e^{-\frac{iT}{n}\Phi_j} a^\dagger - e^{-\frac{iT}{n}\Phi_j} a^\dagger e^{-\frac{iT}{n}\Phi_k} a^\dagger \\ &= e^{-\frac{iT}{n}\Phi_k} a^\dagger e^{\frac{iT}{n}\Phi_k} e^{-\frac{iT}{n}\Phi_{(k+j)}} a^\dagger e^{\frac{iT}{n}\Phi_{(k+j)}} e^{-\frac{iT}{n}\Phi_{k+j}} - \\ &\quad e^{-\frac{iT}{n}\Phi_j} a^\dagger e^{\frac{iT}{n}\Phi_j} e^{-\frac{iT}{n}\Phi_{(k+j)}} a^\dagger e^{\frac{iT}{n}\Phi_{(k+j)}} e^{-\frac{iT}{n}\Phi_{(k+j)}}.\end{aligned} \tag{B.40}$$

Now, terms such as $e^{-\frac{iT}{n}\Phi_j} a^\dagger e^{\frac{iT}{n}\Phi_j}$ can be computed by defining

$$H_{\text{temp}} \equiv -j\phi'(1 + \tilde{\delta}(a^\dagger a - 1)), \tag{B.41}$$

and invoking Heisenberg's equation of motion. Using $[H_{\text{temp}}, a^\dagger] = -j\phi'\tilde{\delta}a^\dagger$ and solving the

differential equation, we obtain

$$e^{-\frac{iT}{n}\Phi_j}a^\dagger e^{\frac{iT}{n}\Phi_j} = e^{-\frac{iT}{n}j\phi'\tilde{\delta}}a^\dagger. \tag{B.42}$$

Using the result of Eq. (B.42) into Eq. (B.40), we have

$$\begin{aligned}[e^{-\frac{iT}{n}\Phi_k}a^\dagger, e^{-\frac{iT}{n}\Phi_j}a^\dagger] &= e^{\frac{-iT}{n}k\phi'\tilde{\delta}}a^\dagger e^{\frac{-iT}{n}(k+j)\phi'\tilde{\delta}}a^\dagger e^{\frac{-iT}{n}\Phi_{(k+j)}} - e^{\frac{-iT}{n}j\phi'\tilde{\delta}}a^\dagger e^{\frac{-iT}{n}(j+k)\phi'\tilde{\delta}}e^{\frac{-iT}{n}\Phi_{(k+j)}} \\ &= e^{\frac{-iT}{n}(j+k)\phi'\tilde{\delta}}(e^{\frac{-iT}{n}k\phi'\tilde{\delta}} - e^{\frac{-iT}{n}j\phi'\tilde{\delta}})(a^\dagger)^2 e^{\frac{-iT}{n}\Phi_{(k+j)}}\end{aligned} \tag{B.43}$$

Using $\phi'\tilde{\delta} = 2\phi$ and expanding Eq. (B.43) to leading order in ϕ, we obtain

$$\begin{aligned}[e^{-\frac{iT}{n}\Phi_k}a^\dagger, e^{-\frac{iT}{n}\Phi_j}a^\dagger] &\approx (1 - \frac{2iT}{n}(j+k)\phi)(\frac{-2iT}{n}\phi)(a^\dagger)^2 e^{\frac{-iT}{n}\Phi_{(k+j)}} \\ &\approx \frac{-2iT}{n}(k-j)\phi(a^\dagger)^2 e^{\frac{-iT}{n}\Phi_{(k+j)}}.\end{aligned} \tag{B.44}$$

Inserting Eq. (B.44) into Eq. (B.39), we obtain

$$[A_k a^\dagger, A_j a^\dagger] = 2\Big(\frac{-iT}{n}\Big)^3(k-j)\phi\mathcal{E}(t_k)\mathcal{E}(t_j)(a^\dagger)^2 e^{\frac{-iT}{n}\Phi_{(k+j)}} + \mathcal{O}(\phi^2). \tag{B.45}$$

A similar calculation shows that

$$[aA_k^\dagger, aA_j^\dagger] = 2\Big(\frac{iT}{n}\Big)^3(k-j)\phi\mathcal{E}(t_k)\mathcal{E}(t_j)e^{\frac{iT}{n}\Phi_{(k+j)}}a^2 + \mathcal{O}(\phi^2). \tag{B.46}$$

Next we compute the cross terms. We first have

$$[A_k a^\dagger, aA_j^\dagger] = \Big(\frac{iT}{n}\Big)^2\mathcal{E}(t_k)\mathcal{E}(t_j)[e^{\frac{-iT}{n}\Phi_k}a^\dagger, ae^{\frac{iT}{n}\Phi_j}], \tag{B.47}$$

with

$$[e^{\frac{-iT}{n}\Phi_k}a^\dagger, ae^{\frac{iT}{n}\Phi_j}] = e^{\frac{-iT}{n}\Phi_k}a^\dagger a e^{\frac{iT}{n}\Phi_j} - ae^{\frac{iT}{n}\Phi_j}e^{\frac{-iT}{n}\Phi_k}a^\dagger$$

$$
\begin{aligned}
&= e^{\frac{-iT}{n}\Phi_k} a^\dagger e^{\frac{iT}{n}\Phi_k} e^{\frac{-iT}{n}\Phi_k} a e^{\frac{iT}{n}\Phi_k} e^{\frac{-iT}{n}\Phi_k} e^{\frac{iT}{n}\Phi_j} - a e^{\frac{-iT}{n}\Phi_{(k-j)}} a^\dagger \\
&= a^\dagger a e^{\frac{-iT}{n}\Phi_{(k-j)}} - a e^{\frac{-iT}{n}\Phi_{(k-j)}} a^\dagger e^{\frac{iT}{n}\Phi_{(k-j)}} e^{\frac{-iT}{n}\Phi_{(k-j)}} \\
&= (a^\dagger a - a a^\dagger e^{\frac{-2iT}{n}\phi(k-j)}) e^{\frac{-iT}{n}\Phi_{(k-j)}} \\
&= (a^\dagger a (1 - e^{\frac{-2iT}{n}\phi(k-j)}) - e^{\frac{-2iT}{n}\phi(k-j)}) e^{\frac{-iT}{n}\Phi_{(k-j)}}. \qquad \text{(B.48)}
\end{aligned}
$$

Expanding the term proportional to $a^\dagger a$ to leading order in ϕ, Eq. (B.48) becomes

$$
[e^{\frac{-iT}{n}\Phi_k} a^\dagger, a e^{\frac{iT}{n}\Phi_j}] \approx \Big(\frac{2iT}{n} \phi a^\dagger a (k-j) - e^{\frac{-2iT}{n}\phi(k-j)} \Big) e^{\frac{-iT}{n}\Phi_{(k-j)}}. \qquad \text{(B.49)}
$$

Inserting Eq. (B.49) into Eq. (B.47), we obtain

$$
\begin{aligned}
[A_k a^\dagger, a A_j^\dagger] \approx 2 \Big(\frac{iT}{n} \Big)^3 \phi(k-j) \mathcal{E}(t_k) \mathcal{E}(t_j) a^\dagger a e^{\frac{-iT}{n}\Phi_{(k-j)}} - \qquad \text{(B.50)} \\
\Big(\frac{iT}{n} \Big)^2 \mathcal{E}(t_k) \mathcal{E}(t_j) e^{\frac{-2iT}{n}\phi(k-j)} e^{\frac{-iT}{n}\Phi_{(k-j)}}. \qquad \text{(B.51)}
\end{aligned}
$$

The last commutator to compute is

$$
[a A_k^\dagger, A_j a^\dagger] = \Big(\frac{-iT}{n} \Big)^2 \mathcal{E}(t_k) \mathcal{E}(t_j) [a e^{\frac{iT}{n}\Phi_k}, e^{\frac{-iT}{n}\Phi_j a^\dagger}], \qquad \text{(B.52)}
$$

with

$$
\begin{aligned}
[a e^{\frac{iT}{n}\Phi_k}, e^{\frac{-iT}{n}\Phi_j} a^\dagger] &= a e^{\frac{iT}{n}\Phi_{(k-j)}} a^\dagger - e^{\frac{-iT}{n}\Phi_j} a^\dagger a e^{\frac{iT}{n}\Phi_k} \\
&= e^{\frac{iT}{n}\Phi_{(k-j)}} e^{\frac{-iT}{n}\Phi_{(k-j)}} a e^{\frac{iT}{n}\Phi_{(k-j)}} a^\dagger - e^{\frac{iT}{n}\Phi_{(k-j)}} e^{\frac{-iT}{n}\Phi_k} a^\dagger e^{\frac{iT}{n}\Phi_k} e^{\frac{-iT}{n}\Phi_k} a e^{\frac{iT}{n}\Phi_k} \\
&= e^{\frac{iT}{n}\Phi_{(k-j)}} (a^\dagger a (e^{\frac{2iT}{n}\phi(k-j)} - 1) + e^{\frac{2iT}{n}\phi(k-j)}) \\
&\approx e^{\frac{iT}{n}\Phi_{(k-j)}} \Big(\frac{2iT}{n} \phi(k-j) a^\dagger a + e^{\frac{2iT}{n}\phi(k-j)} \Big) \qquad \text{(B.53)}
\end{aligned}
$$

so that

$$[aA_k^\dagger, A_j a^\dagger] \approx 2\Big(\frac{iT}{n}\Big)^3 \phi(k-j)\mathcal{E}(t_k)\mathcal{E}(t_j)e^{\frac{iT}{n}\Phi_{(k-j)}}a^\dagger a + \tag{B.54}$$

$$\Big(\frac{-iT}{n}\Big)^2 \mathcal{E}(t_k)\mathcal{E}(t_j)e^{\frac{iT}{n}\Phi_{(k-j)}}e^{\frac{2iT}{n}\phi(k-j)} \tag{B.55}$$

Inserting Eqs. (B.45), (B.46), (B.51) and (B.55) into Eq. (B.37), we have

$$\begin{aligned} D(A_k + A_j) = D(A_k)D(A_j)\exp\Big[&- \frac{1}{2}\Big\{2\Big(\frac{-iT}{n}\Big)^3 (k-j)\phi\mathcal{E}(t_k)\mathcal{E}(t_j)(a^\dagger)^2 e^{\frac{-iT}{n}\Phi_{(k+j)}} \\ &+ 2\Big(\frac{iT}{n}\Big)^3 (k-j)\phi\mathcal{E}(t_k)\mathcal{E}(t_j)e^{\frac{iT}{n}\Phi_{(k+j)}}a^2 \qquad \text{(B.56)} \\ &- 2\Big(\frac{iT}{n}\Big)^3 \phi(k-j)\mathcal{E}(t_k)\mathcal{E}(t_j)a^\dagger a e^{\frac{-iT}{n}\Phi_{(k-j)}} \\ &+ \Big(\frac{iT}{n}\Big)^2 \mathcal{E}(t_k)\mathcal{E}(t_j)e^{\frac{-2iT}{n}\phi(k-j)}e^{\frac{-iT}{n}\Phi_{(k-j)}} \qquad \text{(B.57)} \\ &- 2\Big(\frac{iT}{n}\Big)^3 \phi(k-j)\mathcal{E}(t_k)\mathcal{E}(t_j)e^{\frac{iT}{n}\Phi_{(k-j)}}a^\dagger a \\ &- \Big(\frac{-iT}{n}\Big)^2 \mathcal{E}(t_k)\mathcal{E}(t_j)e^{\frac{iT}{n}\Phi_{(k-j)}}e^{\frac{2iT}{n}\phi(k-j)}\Big\}\Big]. \qquad \text{(B.58)} \end{aligned}$$

Since the product $D(A_n)D(A_{n-1})\cdots D(A_2)D(A_1)$ will produce sums in the exponents proportional to n^2, all terms in Eq. (B.58) proportional to $(\frac{T}{n})^3$ will vanish in the limit where $n \to \infty$. Hence Eq. (B.58) simplifies to

$$D(A_k + A_j) = D(A_k)D(A_j) \tag{B.59}$$

$$\begin{aligned} &\exp\Big[i\mathcal{E}(t_k)\mathcal{E}(t_j)\Big(\frac{T}{n}\Big)^2\Big\{\frac{e^{\frac{-2iT}{n}\phi(k-j)}e^{\frac{-iT}{n}\Phi_{(k-j)}} - e^{\frac{2iT}{n}\phi(k-j)}e^{\frac{iT}{n}\Phi_{(k-j)}}}{2i}\Big\}\Big] \\ &= D(A_k)D(A_j)\exp\Big[-i\mathcal{E}(t_k)\mathcal{E}(t_j)\Big(\frac{T}{n}\Big)^2 \sin\Big(\frac{T}{n}\overline{\Phi}_{(k-j)}\Big)\Big], \end{aligned} \tag{B.60}$$

where $\overline{\Phi}_{(k-j)}$ is defined as

$$\overline{\Phi}_{(k-j)} \equiv (k-j)(\phi - \omega_+ + 2\phi a^\dagger a). \tag{B.61}$$

We thus have that the product of the modified displacement operators is given by

$$D(A_k)D(A_j) = D(A_k + A_j)\text{exp}\Big[i\mathcal{E}(t_k)\mathcal{E}(t_j)\Big(\frac{T}{n}\Big)^2 \sin\Big(\frac{T}{n}\overline{\Phi}_{(k-j)}\Big)\Big]. \tag{B.62}$$

We now wish to compute

$$P_n \equiv D(A_n)D(A_{n-1})D(A_{n-2})\cdots D(A_2)D(A_1). \tag{B.63}$$

First notice that for any operators A and B, to leading order $e^A e^B = e^B e^A e^{[A,B]}$. Now for terms of the form $\text{exp}\Big[i\mathcal{E}(t_k)\mathcal{E}(t_j)\Big(\frac{T}{n}\Big)^2 \sin\Big(\frac{T}{n}\overline{\Phi}_{(k-j)}\Big)\Big]D(\sum_m A_m)$, the commutator of the two exponents will be proportional to $(\frac{T}{n})^3$ which will vanish in the limit where $n \to \infty$. Hence, we can commute all terms $\text{exp}\Big[i\mathcal{E}(t_k)\mathcal{E}(t_j)\Big(\frac{T}{n}\Big)^2 \sin\Big(\frac{T}{n}\overline{\Phi}_{(k-j)}\Big)\Big]$ to the right hand side of P_n. Hence we have

$$P_n = D\Big(\sum_{k=1}^{n} A_k\Big)\text{exp}\Big[i\sum_{k<j}^{n}\frac{T^2}{n^2}\mathcal{E}(t_k)\mathcal{E}(t_j)\sin\Big(\frac{T}{n}\overline{\Phi}_{(k-j)}\Big)\Big]. \tag{B.64}$$

Now, taking the limit where $n \to \infty$ of P_n, we obtain

$$\begin{aligned}\lim_{n\to\infty} P_n &= D\Big(\lim_{n\to\infty}\sum_{k=1}^{n}\Big(\frac{-iT}{n}\Big)\mathcal{E}(t_k)e^{\frac{-iT}{n}\Phi_k}\Big)\text{exp}\Big[i\lim_{n\to\infty}\sum_{k<j}^{n}\frac{T^2}{n^2}\mathcal{E}(t_k)\mathcal{E}(t_j)\sin\Big(\frac{T}{n}\overline{\Phi}_{(k-j)}\Big)\Big] \\ &= D\Big(-i\int_0^T \mathcal{E}(t)e^{-i\Phi t}dt\Big)\text{exp}\Big[-i\int_0^T dt\int_t^T dt'\mathcal{E}(t)\mathcal{E}(t')\sin\Big(\overline{\Phi}(t'-t)\Big)\Big] \end{aligned} \tag{B.65}$$

We conclude that

$$V_+(0,T) = R_+(T)D(A_+)e^{iB_+}, \tag{B.66}$$

where

$$R_+(T) = e^{-iT(\omega_+ - \phi a^\dagger a)a^\dagger a}, \tag{B.67}$$

$$A_+ = -i \int_0^T \mathcal{E}(t) e^{-i\Phi t} dt, \tag{B.68}$$

and

$$B_+ = -\int_0^T dt \int_t^T dt' \mathcal{E}(t)\mathcal{E}(t') \sin\Big(\overline{\Phi}(t'-t)\Big). \tag{B.69}$$

Hence to conclude, the unitary evolution of the Hamiltonian described in Eq. (B.7) is given by

$$V(0,T) = R_+(T) D(A_+) e^{iB_+} e^{-i\tilde{\omega}_a T} |0\rangle\langle 0| + R_-(T) D(A_-) e^{iB_-} e^{i\tilde{\omega}_a T} |1\rangle\langle 1|. \tag{B.70}$$

Recall that in deriving the expression for $V_+(0,T)$, in several steps of the calculation we expanded to leading order in ϕ. Hence the expressions obtained in Eqs. (B.67) to (B.69) are only valid to leading order in ϕ. Hence to leading order in ϕ, we have

$$R_\pm(T) = e^{-iT\omega_\pm a^\dagger a}(1 \pm iT\phi(a^\dagger a)^2) + \mathcal{O}(\phi^2), \tag{B.71}$$

$$A_\pm = -i \int_0^T \mathcal{E}(t) e^{i\omega_\pm t} dt \mp \phi(2a^\dagger a - 1) \int_0^T \mathcal{E}(t) t e^{i\omega_\pm t} dt + \mathcal{O}(\phi^2), \tag{B.72}$$

and

$$\begin{aligned} B_\pm = &\int_0^T dt \int_t^T dt' \mathcal{E}(t)\mathcal{E}(t') \sin\Big(\omega_\pm(t'-t)\Big) \pm \\ &\phi(2a^\dagger a + 1) \int_0^T dt \int_t^T dt' \mathcal{E}(t)\mathcal{E}(t') \cos\Big(\omega_\pm(t'-t)\Big)(t'-t) + \mathcal{O}(\phi^2). \end{aligned} \tag{B.73}$$

B.2.2 Including the Kerr non-linearity

When including the Kerr non-linearity, the Hamiltonian during the control-displacement gate is now given by

$$H(t) = \tilde{\omega}_r a^\dagger a + \tilde{\omega}_a Z + \chi a^\dagger a Z - \phi (a^\dagger a)^2 Z - \frac{K}{2}(a^\dagger a)^2. \tag{B.74}$$

In this case, we have that

$$\begin{aligned} V(0,T) = &\Big(\lim_{n\to\infty} \prod_{j=1}^{n} e^{-i\tilde{t}(\omega_+ - (\phi + \frac{K}{2})a^\dagger a)a^\dagger a} e^{-i\tilde{t}\mathcal{E}(t_j)(a+a^\dagger)} \Big) e^{-i\tilde{\omega}_a T} \left|0\right\rangle \left\langle 0\right| \\ &+ \Big(\lim_{n\to\infty} \prod_{j=1}^{n} e^{-i\tilde{t}(\omega_- + (\phi - \frac{K}{2})a^\dagger a)a^\dagger a} e^{-i\tilde{t}\mathcal{E}(t_j)(a+a^\dagger)} \Big) e^{i\tilde{\omega}_a T} \left|1\right\rangle \left\langle 1\right|, \end{aligned} \tag{B.75}$$

Hence, we see that the analysis of Section B.2.1 is exactly the same, with $\phi \to \phi \pm \frac{K}{2}$ in $R_\pm(T)$, $A_\pm$ and $B_\pm$.

B.2.3 Controlled-displacement gate in the rotating frame

In this section, we consider the same Hamiltonian as in Eq. (B.7) but with a drive term of the form

$$H_d(t) = \mathcal{E}(t) a^\dagger e^{-i\omega_d t} + \mathcal{E}^*(t) a e^{i\omega_d t}. \tag{B.76}$$

We will go into the rotating frame of both the qubit and the cavity. We define

$$H_s' = \tilde{\omega}_r a^\dagger a + \tilde{\omega}_a Z. \tag{B.77}$$

With $U(t) = e^{iH_s' t}$ and applying the transformation $U(t)H(t)U^{-1}(t) - iU(t)\frac{\partial}{\partial t}U^{-1}(t) \equiv H_R(t)$ to the Hamiltonian in Eq. (B.7) with $H_d(t)$ given in Eq. (B.76), we obtain

$$H_R(t) = \chi Z a^\dagger a - \phi(a^\dagger a)^2 Z + \mathcal{E}(t) a^\dagger e^{i(\tilde{\omega}_r - \omega_d)t} + \mathcal{E}^*(t) a e^{-i(\tilde{\omega}_r - \omega_d)t} \tag{B.78}$$

The frequency dependence in the drive term of Eq. (B.78) can be eliminated by choosing $\omega_d = \tilde{\omega}_r$. Again, we wish to compute the unitary evolution

$$V_R(0,T) = \mathcal{T} e^{-i\int_0^T H_R(t')dt'}. \tag{B.79}$$

Using the Suzuki-Trotter decomposition

$$\begin{aligned} V_R(0,T) = &\left(\lim_{n\to\infty} \prod_{j=1}^{n} e^{-i\tilde{t}(\omega_+ - \phi a^\dagger a)a^\dagger a} e^{-i\tilde{t}(\mathcal{E}(t_j)a^\dagger + \mathcal{E}^*(t_j)a)} \right) |0\rangle\langle 0| \\ &+ \left(\lim_{n\to\infty} \prod_{j=1}^{n} e^{-i\tilde{t}(\omega_- + \phi a^\dagger a)a^\dagger a} e^{-i\tilde{t}(\mathcal{E}(t_j)a^\dagger + \mathcal{E}^*(t_j)a)} \right) |1\rangle\langle 1|, \end{aligned} \tag{B.80}$$

where now we have that $\omega_\pm = \pm\chi$.

Comparing Eq. (B.80) to Eq. (B.13), we see that we can follow the same steps as in Section B.2.1 by using the new values for $\omega_\pm$ and replacing $\mathcal{E}$ by $\mathcal{E}^*$ in the conjugate expressions. Doing so, we obtain

$$V_R(0,T) = R_+(T) D(A_+) e^{iB_+} |0\rangle\langle 0| + R_-(T) D(A_-) e^{iB_-} |1\rangle\langle 1|, \tag{B.81}$$

where

$$R_\pm(T) = e^{-iT\omega_\pm a^\dagger a}(1 \pm iT\phi(a^\dagger a)^2), \tag{B.82}$$

$$A_{\pm} = -i \int_0^T \mathcal{E}(t) e^{i\omega_{\pm} t} dt \mp \phi(2a^{\dagger}a - 1) \int_0^T \mathcal{E}(t) t e^{i\omega_{\pm} t} dt + \mathcal{O}(\phi^2), \tag{B.83}$$

$$\begin{aligned} B_{\pm} = &-\frac{i}{2} \int_0^T dt \int_t^T dt' \Big(\mathcal{E}^*(t') \mathcal{E}(t) e^{i\omega_{\pm}(t'-t)} - \mathcal{E}(t') \mathcal{E}^*(t) e^{-i\omega_{\pm}(t'-t)} \Big) \\ &\pm \frac{\phi(2a^{\dagger}a + 1)}{2} \int_0^T dt \int_t^T dt' \Big(\mathcal{E}^*(t') \mathcal{E}(t) e^{i\omega_{\pm}(t'-t)} + \mathcal{E}(t') \mathcal{E}^*(t) e^{i\omega_{\pm}(t'-t)} \Big)(t' - t) + \mathcal{O}(\phi^2). \end{aligned} \tag{B.84}$$

B.2.4 Effects of the non-linear dispersive shift and Kerr term on the unitary evolution of the qubit-cavity system

In this section we will show that regardless of the chosen pulse shape (even with numerical tools such as optimal control), to leading order in $\phi_{\pm}$, the changes to the unitary evolution of the qubit-cavity system due to the non-linear dispersive shift and Kerr terms cannot be completely removed for time scales $T \ll \frac{1}{\phi_{\pm}}$. Using Eqs. (B.67), (B.72) and (B.73), we can write (for instance choosing the terms affecting the $|0\rangle\langle 0|$

$$R_+(T) D(A_+) = e^{-i\omega_+ T a^{\dagger}a} e^{i\omega_+ T \phi_+ (a^{\dagger}a)^2} e^{I_1 a^{\dagger} - I_1^* a - I_2 \phi_+ (2a^{\dagger}a - 1)a^{\dagger} + I_2^* \phi_+ a (2a^{\dagger}a - 1)}, \tag{B.85}$$

where we define

$$I_1 \equiv -i \int_0^T \mathcal{E}(t) e^{i\omega_+ t} dt, \tag{B.86}$$

and

$$I_2 \equiv \int_0^T \mathcal{E}(t) t e^{i\omega_+ t} dt. \tag{B.87}$$

Let $A = i\omega_+ T\phi_+ (a^\dagger a)^2$ and $B = I_1 a^\dagger - I_1^* a - I_2\phi_+(2a^\dagger a - 1)a^\dagger + I_2^*\phi_+ a(2a^\dagger a - 1)$. Using the Baker-Campbell-Hausdorff lemma and keeping only leading order terms in ϕ_+, we have

$$e^A e^B = e^{A+B+\frac{1}{2}[A,B]-\frac{1}{12}[B[A,B]]-\frac{1}{720}[B,[B,[B,[B,A]]]]}. \tag{B.88}$$

Computing the commutators, we find

$$e^A e^B = e^{\frac{1}{6}i\omega_+ T\phi_+(-\frac{1}{5}|I_1|^4+|I_1|^2)} e^L \tag{B.89}$$

where

$$\begin{aligned} L = [&(I_2\phi_+ + (1+\frac{1}{2}i\omega_+ T\phi_+)I_1)a^\dagger - (I_2^\dagger\phi_+ + \\ &(1-\frac{1}{2}i\omega_+ T\phi_+)I_1^\dagger)a + \frac{1}{6}i\omega_+ T\phi_+(I_1^2(a^\dagger)^2 + 4|I_1|^2 \\ &a^\dagger a + (I^\dagger)^2 a^2) + i\omega_+ T\phi_+(I_1 a^\dagger a^\dagger a + I_1^\dagger a^\dagger aa) - 2I_2\phi_+ a^\dagger aa^\dagger + 2I_2^\dagger\phi_+ aa^\dagger a + \omega_+ T\phi_+(a^\dagger a)^2] \end{aligned}$$

Notice that the last term in Eq. (B.89) is proportional to $(a^\dagger a)^2$ and independent of the pulse shape (performing the same calculation as above including the B_+ term will not change the conclusion). The terms dependent on the pulse shape are expressed as lower powers of a and $a^\dagger$. One cannot eliminate all terms proportional to ϕ_+ unless one chooses a time scale comparable to $\frac{1}{\phi_+}$. For time scales on the order of $\frac{1}{\phi_+}$, higher order terms in ϕ_+ will be relevant and therefore it might possible to choose pulse (with numerical techniques such as optimal control) which can eliminate effects from the non-linear dispersive shift and Kerr terms. However using the parameters of Table 4.2, $\frac{1}{\phi_+}$ is roughly four orders of magnitude longer than the chosen time scale ($T = \frac{\pi}{\chi}$) of our protocol. For such long time scales, effects due to photon loss, damping and dephasing would render the protocol impractical.

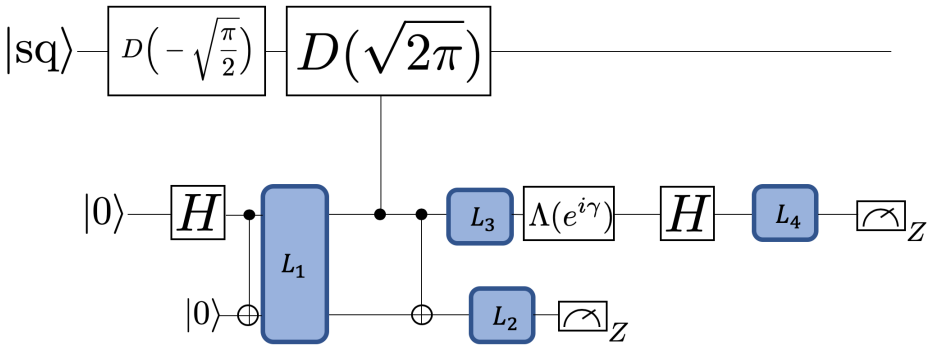


Figure B.2: Illustration of error locations (blue boxes) before and after the controlled displacement gate, where Pauli errors were added based on their corresponding error probability polynomials (computed from the depolarizing channel described in Section 4.7.2). Errors during the controlled-displacement gate were simulated using the master equation described in Section 4.7.2.

B.2.5 Details of numerical simulation

From the depolarizing noise model described in Section 4.7.2, we computed error probabilities for all Pauli errors arising from a single fault at the locations indicated by the blue boxes in Fig. B.2. For instance, the probability of an $I \otimes Z$ error at location L_1 is given by

$$P_{IZ} = \frac{p}{15}\Big(P1_X P1_Z + P1_X(1 - P1_Z) + (1 - P1_X)(1 - P1_Z)\Big) + (1-p)(1 - P1_X)P1_Z, \tag{B.90}$$

where

$$P1_X = \frac{p}{15}, \tag{B.91}$$

during the first round, and

$$P1_X = \frac{2p}{3}, \tag{B.92}$$

in later rounds. Similarly,

$$P1_Z = \frac{p}{15}(1 - \frac{p}{10}) + \frac{p^2}{450} + (1 - \frac{p}{15})\frac{p}{15}, \tag{B.93}$$

for the first round and

$$P1_Z = \frac{2p}{3}(1 - \frac{p}{10}) + \frac{2p}{3}\frac{p}{30} + (1 - \frac{2p}{3})\frac{p}{15}. \tag{B.94}$$

With a Pauli error added at either locations L_1, L_2, L_3 or L_4, we update the state of the qubit before performing the master equation simulation described in Section 4.7.2. With the analytic error probability for the Pauli error, in addition to the evolution of the state during the master equation, we can compute the total probability of obtaining the output state. As mentioned in Section 4.7.2, such an analysis ignored second order Pauli error events (before and after the controlled displacement gate).

APPENDIX C

DETAILS OF THE CERTIQ FRAMEWORK

C.1 Syntax of Quantum Circuits supported in CertiQ

$$
\begin{aligned}
QProgram &::= RegisterDeclList;\ Circuit \\
RegisterDeclList &::= RegisterDecl(;RegisterDecl)* \\
RegisterDecl &::= Type\ Id\,[\,Int\,] \\
Type &::= \textbf{Qbit} \mid \textbf{Cbit} \\
Circuit &::= Inst[;Inst;...] \\
Inst &::= Unitary\_Inst \mid Non\_unitary\ (op)* \mid \\
&\quad\ \textbf{Controlled}\ Unitary\_Inst\ op \\
op &::= Id[Int] \\
Unitary\_Inst &::= Unary\ op \mid Binary\ op1, op2 \mid \\
&\quad\ Custom\ (op)* \\
Unary &::= \textbf{X} \mid \textbf{Y} \mid \textbf{Z} \mid \textbf{Rx}\ Float \mid \textbf{Ry}\ Float \mid \textbf{Rz}\ Float \mid \\
&\quad\ \textbf{U1}\ Float \mid \textbf{U2}\ (Float)* \mid \textbf{U3}\ (Float)* \\
Binary &::= \textbf{CX} \mid \textbf{CZ} \mid \textbf{CY} \mid \textbf{iSWAP} \mid \textbf{MS} \mid \textbf{SWAP} \\
Custom &::= \textbf{Custom}\ Matrix \\
Non\_unitary &::= \textbf{Measure} \mid \textbf{Barrier} \mid \textbf{Reset}
\end{aligned}
$$

Figure C.1: Syntax of the quantum programs in CertiQ.

C.2 Shortcut Lemmas

The CertiQ verifies can only check the equivalence of circuits with fixed length due to the limitation of SMT solvers. When verifying the equivalence of circuits with a variable length (which typically happens when loops are involved), SMT solvers will fail. To solve this problem, we add a set of lemmas to specify behaviors about the symbolic execution results of an arbitrary list. For example, we add an lemma stating that if every gate in the two lists is equal, then the symbolic execution result must be equivalent. By these shortcut lemmas, CertiQ verifier can avoid executing a variable length circuit and run efficiently. Consider a very simple optimization: if the last two gates of

a circuit are SWAP gates on the same qubit pair, then delete them. Z3 will quickly prove the correctness of this optimization by the above shortcut lemma and SWAP rules defined in Fig. 5.7.

C.3 The CertiQ Interface

CertiQ provides a Python-like interface for users to write annotated compiler pass implementations. This interface language supports branch statements and loops with loop annotations. Loops annotation will provide information for the parser to generate verification conditions. The compiler passes written in this language is very close to the executable version, except that the library invocations have to be linked with the wrapper libraries for Qiskit such that linked implementation can be integrated into Qiskit and executed on real hardware.

We highlight important syntax of this interface language as follows:

$$
\begin{aligned}
Program &::= StmtList \\
stmt &::= \texttt{If}\ Expr : Stmt\ \texttt{else} : Stmt \\
StmtList &::= Stmt1 Stmt2 \cdots Stmtn \\
&\quad \mid \texttt{for}\ Iterator\ \texttt{in}\ List\ LoopAnno\ \texttt{:}\ StmtList \\
&\quad \mid \texttt{While}\ Expr\ LoopAnno\ \texttt{:}\ StmtList \\
&\quad \mid \cdots \\
LoopAnno &::= \#@Id\texttt{:}Expr[Id\texttt{:}Expr] \\
FuncDef &::= \texttt{def}\ Id(ParaList)\ \texttt{:}\ StmtList
\end{aligned}
$$

REFERENCES

[1] Cirq: A python framework for creating, editing, and invoking noisy intermediate scale quantum (NISQ) circuits. https://github.com/quantumlib/Cirq, 2018.

[2] Dorit Aharonov and Michael Ben-Or. Fault-tolerant quantum computation with constant error. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 176–188. ACM, 1997.

[3] Tameem Albash and Daniel A. Lidar. Adiabatic quantum computation. *Reviews of Modern Physics*, 90(1), Jan 2018.

[4] Victor V Albert, Kyungjoo Noh, Kasper Duivenvoorden, Dylan J Young, R T Brierley, Philip Reinhold, Christophe Vuillot, Linshu Li, Chao Shen, S M Girvin, Barbara M Terhal, and Liang Jiang. Performance and structure of single-mode bosonic codes. *Phys. Rev. A*, 97(3):32346, mar 2018.

[5] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, Luciano Bello, Yael Ben-Haim, David Bucher, Francisco Jose Cabrera-Hernández, Jorge Carballo-Franquis, Adrian Chen, Chun-Fu Chen, Jerry M. Chow, Antonio D. Córcoles-Gonzales, Abigail J. Cross, Andrew Cross, Juan Cruz-Benito, Chris Culver, Salvador De La Puente González, Enrique De La Torre, Delton Ding, Eugene Dumitrescu, Ivan Duran, Pieter Eendebak, Mark Everitt, Ismael Faro Sertage, Albert Frisch, Andreas Fuhrer, Jay Gambetta, Borja Godoy Gago, Juan Gomez-Mosquera, Donny Greenberg, Ikko Hamamura, Vojtech Havlicek, Joe Hellmers, Łukasz Herok, Hiroshi Horii, Shaohan Hu, Takashi Imamichi, Toshinari Itoko, Ali Javadi-Abhari, Naoki Kanazawa, Anton Karazeev, Kevin Krsulich, Peng Liu, Yang Luh, Yunho Maeng, Manoel Marques, Francisco Jose Martín-Fernández, Douglas T. McClure, David McKay, Srujan Meesala, Antonio Mezzacapo, Nikolaj Moll, Diego Moreda Rodríguez, Giacomo Nannicini, Paul Nation, Pauline Ollitrault, Lee James O'Riordan, Hanhee Paik, Jesús Pérez, Anna Phan, Marco Pistoia, Viktor Prutyanov, Max Reuter, Julia Rice, Abdón Rodríguez Davila, Raymond Harry Putra Rudy, Mingi Ryu, Ninad Sathaye, Chris Schnabel, Eddie Schoute, Kanav Setia, Yunong Shi, Adenilton Silva, Yukio Siraichi, Seyon Sivarajah, John A. Smolin, Mathias Soeken, Hitomi Takahashi, Ivano Tavernelli, Charles Taylor, Pete Taylour, Kenso Trabing, Matthew Treinish, Wes Turner, Desiree Vogt-Lee, Christophe Vuillot, Jonathan A. Wildstrom, Jessica Wilson, Erick Winston, Christopher Wood, Stephen Wood, Stefan Wörner, Ismail Yunus Akhalwaya, and Christa Zoufal. Qiskit: An open-source framework for quantum computing, 2019.

[6] Panos Aliferis and Andrew W. Cross. Subsystem fault tolerance with the bacon-shor code. *Phys. Rev. Lett.*, 98:220502, May 2007.

[7] Panos Aliferis, Daniel Gottesman, and John Preskill. Quantum accuracy threshold for concatenated distance-3 codes. *Quantum Info. Comput.*, 6(2):97–165, March 2006.

[8] J. Alonso, F. M. Leupold, Z. U. Soler, M. Fadel, M. Marinelli, B. C. Keitch, V. Negnevitsky, and J. P. Home. Generation of large coherent states by bang-bang control of a trapped-ion oscillator. *Nature communications*, 7(11243), 2016.

[9] Matthew Amy. Towards Large-scale Functional Verification of Universal Quantum Circuits. *Electronic Proceedings in Theoretical Computer Science*, 287:1–21, jan 2019.

[10] Matthew Amy, Martin Roetteler, and Krysta M. Svore. Verified Compilation of Space-Efficient Reversible Circuits. In *Computer Aided Verification*, pages 3–21. International Conference on Computer Aided Verification, 2017.

[11] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[12] Ben Q. Baragiola, Giacomo Pantaleoni, Rafael N. Alexand er, Angela Karanjai, and Nicolas C. Menicucci. All-Gaussian universality and fault tolerance with the Gottesman-Kitaev-Preskill code. *arXiv e-prints*, page arXiv:1903.00012, Feb 2019.

[13] F Bloch. Nuclear Induction. *Phys. Rev.*, 70(7-8):460–474, oct 1946.

[14] Maxime Boissonneault, J. M. Gambetta, and Alexandre Blais. Dispersive regime of circuit qed: Photon-dependent qubit dephasing and relaxation rates. *Phys. Rev. A*, 79:013819, Jan 2009.

[15] A. Borzı‘, J. Salomon, and S. Volkwein. Formulation and numerical solution of finite-level quantum optimal control problems. *Journal of Computational and Applied Mathematics*, 216(1):170–197, June 2008.

[16] Teresa Brecht, Wolfgang Pfaff, Chen Wang, Yiwen Chu, Luigi Frunzio, Michel H Devoret, and Robert J Schoelkopf. Multilayer microwave integrated quantum circuits for scalable quantum computing. *npj Quantum Information*, 2(16002), 2016.

[17] Kenneth R Brown, Jungsang Kim, and Christopher Monroe. Co-designing a scalable quantum computer with trapped atomic ions. *npj Quantum Information*, 2:16034, 2016.

[18] Juan Carlos Garcia-Escartin and Pedro Chamorro-Posada. Equivalent Quantum Circuits. Technical report, Universidad de Valladolid, Dpto. Teoria de la Senal e Ing., 2011.

[19] Christopher Chamberland and Michael E. Beverland. Flag fault-tolerant error correction with arbitrary distance codes. *Quantum*, 2:53, February 2018.

[20] Christopher Chamberland and Andrew Cross. Fault-tolerant magic state preparation with flag qubits. *arXiv:quant-ph/1811.00566*, 2018.

[21] Christopher Chamberland, Tomas Jochym-O'Connor, and Raymond Laflamme. Thresholds for universal concatenated quantum codes. *Phys. Rev. Lett.*, 117:010501, 2016.

[22] Christopher Chamberland, Tomas Jochym-O'Connor, and Raymond Laflamme. Overhead analysis of universal concatenated quantum codes. *Phys. Rev. A*, 95:022313, Feb 2017.

[23] Christopher Chamberland and Pooya Ronagh. Deep neural decoders for near term fault-tolerant experiments. *Quantum Science and Technology*, 3(4):044002, jul 2018.

[24] Rui Chao and Ben W. Reichardt. Fault-tolerant quantum computation with few qubits. *npj Quantum Information*, 4, 2018.

[25] Rui Chao and Ben W. Reichardt. Quantum error correction with only two extra qubits. *Phys. Rev. Lett.*, 121:050502, Aug 2018.

[26] Andrew M. Childs, Eddie Schoute, and Cem M. Unsal. Circuit transformations for quantum architectures. 2019.

[27] Kevin S. Chou, Jacob Z. Blumoff, Christopher S. Wang, Philip C. Reinhold, Christopher J. Axline, Yvonne Y. Gao, L. Frunzio, M. H. Devoret, Liang Jiang, and R. J. Schoelkopf. Deterministic teleportation of a quantum gate between two logical qubits. *Nature*, 561(7723):1476–4687, 2018.

[28] M. J. Chow. *Quantum Information Processing with Superconducting Qubits*. PhD thesis, New Haven, CT, USA, 2010. AAINQ98874.

[29] J. I. Cirac and P. Zoller. Quantum computations with cold trapped ions. *Physical Review Letters*, 74(20):4091–4094, 1995.

[30] The Coq Development Team. *The Coq Reference Manual, version 8.4*, August 2012. Available electronically at http://coq.inria.fr/doc.

[31] A. W. Cross, L. S. Bishop, J. A. Smolin, and J. M. Gambetta. Open Quantum Assembly Language. *ArXiv e-prints*, July 2017.

[32] Andrew W. Cross, David P. Divincenzo, and Barbara M. Terhal. A comparative code study for quantum fault tolerance. *Quantum Info. Comput.*, 9(7):541–572, July 2009.

[33] Christopher M. Dawson and Michael Nielsen. The solovay-kitaev algorithm. *Quant. Inf. Comput.*, 6:81–95, 2006.

[34] 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.

[35] P. de Fouquieres, S. G. Schirmer, S. J. Glaser, and Ilya Kuprov. Second order gradient ascent pulse engineering. *Journal of Magnetic Resonance*, 212(2):412–417, October 2011.

[36] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient SMT Solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 4963 LNCS, pages 337–340, 2008.

[37] S. Debnath, N. M. Linke, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe. Demonstration of a small programmable quantum computer with atomic qubits. *Nature*, 536, Aug 2016.

[38] S. Debnath, N. M. Linke, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe. Demonstration of a small programmable quantum computer with atomic qubits. *Nature*, 536:63 EP –, Aug 2016.

[39] Michel H. Devoret and Robert J. Schoelkopf. Superconducting circuits for quantum information: an outlook. *Science*, 339(6124):1169–1174, 2013.

[40] P. Ditz and A. Borzi‘. A cascadic monotonic time-discretized algorithm for finite-level quantum control computation. *Computer Physics Communications*, 178(5):393–399, March 2008.

[41] David P. DiVincenzo. Two-qubit gates are universal for quantum computation. *Phys. Rev. A*, 51:1015–1022, 1995.

[42] David P. DiVincenzo. The physical implementation of quantum computation. *Fortschritte der Physik*, 48(9-11):771783, Sep 2000.

[43] Tom Douce, Damian Markham, Elham Kashefi, Peter van Loock, and Giulia Ferrini. Probabilistic fault-tolerant universal quantum computation and sampling problems in continuous variables. *Phys. Rev. A*, 99:012344, Jan 2019.

[44] Kasper Duivenvoorden, Barbara M. Terhal, and Daniel Weigand. Single-mode displacement sensor. *Phys. Rev. A*, 95:012305, Jan 2017.

[45] Miller Eaton, Rajveer Nehra, and Olivier Pfister. Gottesman-kitaev-preskill state preparation by photon catalysis. *arXiv:quant-ph/1903.01925*, 2019.

[46] Reuven Eitan, Michael Mundt, and David J. Tannor. Optimal control with accelerated convergence: Combining the Krotov and quasi-Newton methods. *Phys. Rev. A*, 83:053426, May 2011.

[47] E. Farhi, J. Goldstone, and S. Gutmann. A Quantum Approximate Optimization Algorithm. *ArXiv e-prints*, November 2014.

[48] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution, 2000.

[49] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21:467, 1982.

[50] C. Fluhmann, T. L. Nguyen, M. Marinelli, V. Negnevitsky, K. Mehta, and J. P. Home. Encoding a qubit in a trapped-ion mechanical oscillator. *Nature*, 566(7745):513–517, 2019.

[51] X. Fu, M. A. Rol, C. C. Bultink, J. van Someren, N. Khammassi, I. Ashraf, R. F. L. Vermeulen, J. C. de Sterke, W. J. Vlothuizen, R. N. Schouten, C. G. Almudever, L. DiCarlo, and K. Bertels. An experimental microarchitecture for a superconducting quantum processor. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17, pages 813–825. ACM, 2017.

[52] X. Fu, M. A. Rol, C. C. Bultink, J. van Someren, N. Khammassi, I. Ashraf, R. F. L. Vermeulen, J. C. de Sterke, W. J. Vlothuizen, R. N. Schouten, C. G. Almudever, L. DiCarlo, and K. Bertels. An experimental microarchitecture for a superconducting quantum processor. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17, pages 813–825, New York, NY, USA, 2017. ACM.

[53] X. Fu, M. A. Rol, C. C. Bultink, J. van Someren, N. Khammassi, I. Ashraf, R. F. L. Vermeulen, J. C. de Sterke, W. J. Vlothuizen, R. N. Schouten, C. G. Almudever, L. DiCarlo, and K. Bertels. A microarchitecture for a superconducting quantum processor. *IEEE Micro*, 38(3):40–47, May 2018.

[54] Kosuke Fukui, Akihisa Tomita, Atsushi Okamoto, and Keisuke Fujii. High-threshold fault-tolerant quantum computation with analog quantum error correction. *Phys. Rev. X*, 8:021054, May 2018.

[55] Jay Gambetta, Alexandre Blais, M. Boissonneault, A. A. Houck, D. I. Schuster, and S. M. Girvin. Quantum trajectory approach to circuit qed: Quantum jumps and the zeno effect. *Phys. Rev. A*, 77:012112, Jan 2008.

[56] Yvonne Y. Gao, Brian J. Lester, Kevin S. Chou, Luigi Frunzio, Michel H. Devoret, Liang Jiang, S. M. Girvin, and Robert J. Schoelkopf. Entanglement of bosonic modes through an engineered exchange interaction. *Nature*, 566(7745):1476–4687, 2019.

[57] G. Giacomo Guerreschi and J. Park. Gate scheduling for quantum algorithms. *ArXiv e-prints*, July 2017.

[58] Philippe Gille and Tamas Szamuely. *Central Simple Algebras and Galois Cohomology*. Cambridge University Press, 2009.

[59] Steven M Girvin. Circuit qed: superconducting qubits coupled to microwave photons. *Quantum Machines: Measurement and Control of Engineered Quantum Systems*, page 113, 2011.

[60] S. Glancy and E. Knill. Error analysis for encoding a qubit in an oscillator. *Phys. Rev. A*, 73:012325, Jan 2006.

[61] 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. *Eur. Phys. J. D*, 69(12):1–24, 2015.

[62] Andrew N. Glaudell, Neil J. Ross, and Jacob M. Taylor. Optimal two-qubit circuits for universal fault-tolerant quantum computation, 2020.

[63] Pranav Gokhale. Github: graph-mapper. https://github.com/singular-value/graph-mapper, 2018.

[64] 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*, MICRO 52, page 266278, New York, NY, USA, 2019. Association for Computing Machinery.

[65] 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*, MICRO '52, pages 266–278, New York, NY, USA, 2019. ACM.

[66] Caroline Gollub, Markus Kowalewski, and Regina de Vivie-Riedle. Monotonic Convergent Optimal Control Theory with Strict Limitations on the Spectrum of Optimized Laser Fields. *Phys. Rev. Lett.*, 101:073002, August 2008.

[67] Daniel Gottesman. Theory of fault-tolerant quantum computation. *Phys. Rev. A*, 57(1):127, 1998.

[68] Daniel Gottesman. An introduction to quantum error correction and fault-tolerant quantum computation. *Proceedings of Symposia in Applied Mathematics*, 68:13–58, 2010.

[69] Daniel Gottesman. Fault-tolerant quantum computation with constant overhead. *Quantum Info. Comput.*, 14(15-16):1338–1372, November 2014.

[70] Daniel Gottesman, Alexei Kitaev, and John Preskill. Encoding a qubit in an oscillator. *Phys. Rev. A*, 64(1):12310, jun 2001.

[71] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219. ACM, 1996.

[72] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219. ACM, 1996.

[73] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Annual ACM Symposium on Theory of Computing*, pages 212–219. ACM, 1996.

[74] Gian Giacomo Guerreschi and Jongsoo Park. Two-step approach to scheduling quantum circuits. *Quantum Science and Technology*, 3(4):045003, jul 2018.

[75] G. Hao Low and I. L. Chuang. Optimal Hamiltonian Simulation by Quantum Signal Processing. *ArXiv e-prints*, June 2016.

[76] Aram W. Harrow, Benjamin Recht, and Isaac L. Chuang. Efficient discrete approximations of quantum gates. *Journal of Mathematical Physics*, 43(9):44454451, Sep 2002.

[77] Reinier W. Heeres, Philip Reinhold, Nissim Ofek, Luigi Frunzio, Liang Jiang, Michel H. Devoret, and Robert J. Schoelkopf. Implementing a universal gate set on a logical qubit encoded in an oscillator. *Nature Communications*, 8(1):94, 2017.

[78] Kesha Hietala, Robert Rand, Shih-Han Hung, Xiaodi Wu, and Michael Hicks. Verified Optimization in a Quantum Intermediate Representation. *arxiv*, apr 2019.

[79] Ling Hu, Yuwei Ma, Weizhou Cai, Xianghao Mu, Yuan Xu, Weiting Wang, Yukai Wu, Haiyan Wang, Yipu Song, Changling Zou, S. M. Girvin, L-M. Duan, and Luyan Sun. Demonstration of quantum error correction and universal gate set on a binomial bosonic logical qubit. *arXiv e-prints*, page arXiv:1805.09072, May 2018.

[80] Thomas Hner, Damian S Steiger, Krysta Svore, and Matthias Troyer. A software methodology for compiling quantum programs. *Quantum Science and Technology*, 3(2):020501, 2018.

[81] 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.

[82] 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. Computing Frontiers 2014: Best Papers.

[83] Zhang Jiang, Jarrod McClean, Ryan Babbush, and Hartmut Neven. Majorana loop stabilizer codes for error correction of fermionic quantum simulations, 2018.

[84] J.R. Johansson, P.D. 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.

[85] J.R. Johansson, P.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.

[86] J.R. Johansson, P.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.

[87] N. C. Jones, R. Van Meter, A. G. Fowler, P. L. McMahon, J. Kim, T. D. Ladd, and Y. Yamamoto. Layered Architecture for Quantum Computing. *Physical Review X*, 2(3):031007, July 2012.

[88] 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*, 549:242 EP –, Sep 2017.

[89] B. E. Kane. A silicon-based nuclear spin quantum computer. *Nature*, 393:133 EP –, May 1998. Article.

[90] Eliot Kapit. Hardware-efficient and fully autonomous quantum error correction in superconducting circuits. *Physical Review Letters*, 116(15), Apr 2016.

[91] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. In *SIAM Journal on Scientific Computing*, volume 20, 02 1970.

[92] 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, February 2005.

[93] Navin Khaneja, Timo Reiss, Cindie Kehlet, Thomas Schulte-Herbrggen, 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.

[94] A. Yu. Kitaev, A. H. Shen, and M. N. Vyalyi. *Classical and Quantum Computation*. American Mathematical Society, USA, 2002.

[95] Alexei Y. Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191–1249, 1997.

[96] A.Yu. Kitaev. Fault-tolerant quantum computation by anyons. *Annals of Physics*, 303(1):2 – 30, 2003.

[97] P. V. Klimov, J. Kelly, Z. Chen, M. Neeley, A. Megrant, B. Burkett, R. Barends, K. Arya, B. Chiaro, Yu Chen, A. Dunsworth, A. Fowler, B. Foxen, C. Gidney, M. Giustina, R. Graff, T. Huang, E. Jeffrey, Erik Lucero, J. Y. Mutus, O. Naaman, C. Neill, C. Quintana, P. Roushan, Daniel Sank, A. Vainsencher, J. Wenner, T. C. White, S. Boixo, R. Babbush, V. N. Smelyanskiy, H. Neven, and John M. Martinis. Fluctuations of energy-relaxation times in superconducting qubits. *Phys. Rev. Lett.*, 121:090502, Aug 2018.

[98] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Fast and efficient exact synthesis of single qubit unitaries generated by clifford and t gates, 2012.

[99] Emanuel Knill. Fault-tolerant postselected quantum computation: schemes. *arXiv:quant-ph/0402171*, 2004.

[100] Emanuel Knill. Quantum computing with realistically noisy devices. *Nature*, 434(7029):39–44, 2005.

[101] Emanuel Knill and Raymond Laflamme. Theory of quantum error-correcting codes. *Phys. Rev. A*, 55:900–911, 1997.

[102] J. Koch, T. M. Yu, J. Gambetta, A. A. Houck, D. I. Schuster, J. Majer, A. Blais, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. Introducing the Transmon: a new superconducting qubit from optimizing the Cooper Pair Box. *eprint arXiv:cond-mat/0703002*, February 2007.

[103] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, , and W. D. Oliver. A quantum engineer's guide to superconducting qubits. 2019.

[104] B. P. Lanyon, J. D. Whitfield, G. G. Gillett, M. E. Goggin, M. P. Almeida, I. Kassal, J. D. Biamonte, M. Mohseni, B. J. Powell, M. Barbieri, A. Aspuru-Guzik, and A. G. White. Towards quantum chemistry on a quantum computer. *Nature Chemistry*, 2:106–111, February 2010.

[105] Chris Lattner and Vikram Adve. LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation. Technical report, UIUC, 2003.

[106] A. Lemmer, A. Bermudez, and M. B. Plenio. Driven geometric phase gates with trapped ions. *New Journal of Physics*, 15(8):083001, August 2013.

[107] Xavier Leroy. Formal verification of a realistic compiler. *Communications of the ACM*, 52(7):107–115, 2009.

[108] 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*, 95:042318, Apr 2017.

[109] Shusen Liu, Xin Wang, Li Zhou, Ji Guan, Yinan Li, Yang He, Runyao Duan, and Mingsheng Ying. Q— SI?: A quantum programming environment. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11180 LNCS, pages 133–164. Springer Verlag, 2018.

[110] Seth Lloyd. Universal quantum simulators. *Science*, 237:1073–1077, 1996.

[111] Nuno P. Lopes, David Menendez, Santosh Nagarakatte, and John Regehr. Provably correct peephole optimizations with alive. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 15, page 2232, New York, NY, USA, 2015. Association for Computing Machinery.

[112] Yao Lu, S. Chakram, Ngainam Leung, Nathan Earnest, Ravi Naik, Ziwen Huang, Peter Groszkowski, Eliot Kapit, Jens Koch, and David Schuster. Universal stabilization of a parametrically coupled qubit. *Physical Review Letters*, 119, 07 2017.

[113] Nancy Lynch and Frits Vaandrager. Forward and backward simulations. *Information and Computation*, 121(2):214–233, 1995.

[114] Yvon Maday and Gabriel Turinici. New formulations of monotonically convergent quantum control algorithms. *The Journal of Chemical Physics*, 118(18):8191–8196, May 2003.

[115] D. Maslov, G. W. Dueck, and D. M. Miller. Toffoli network synthesis with templates. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(6):807–817, June 2005.

[116] D. Maslov, G. W. Dueck, D. M. Miller, and C. Negrevergne. Quantum circuit simplification and level compaction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(3):436–444, March 2008.

[117] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Aln Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, 2016.

[118] 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 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*, sep 2018.

[119] Microsoft. F*, 2016.

[120] D. M. Miller and Z. Sasanian. Lowering the quantum gate cost of reversible circuits. In *2010 53rd IEEE International Midwest Symposium on Circuits and Systems*, pages 260–263, Aug 2010.

[121] Z. K. Minev, K. Serniak, I. M. Pop, Z. Leghtas, K. Sliwa, M. Hatridge, L. Frunzio, R. J. Schoelkopf, and M. H. Devoret. Planar multilayer circuit quantum electrodynamics. *Phys. Rev. Applied*, 5:044021, Apr 2016.

[122] 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*, ASPLOS '19, pages 1015–1029, New York, NY, USA, 2019. ACM.

[123] Prakash Murali, Jonathan M. Baker, Ali Javadi-Abhari, Frederic T. Chong, and Margaret Martonosi. Noise-Adaptive Compiler Mappings for Noisy Intermediate-Scale Quantum Computers. pages 1015–1029. Association for Computing Machinery (ACM), 2019.

[124] Prakash Murali, Norbert Matthias Linke, Margaret Martonosi, Ali Javadi Abhari, Nhung Hong Nguyen, and Cinthia Huerta Alderete. Full-stack, Real-system Quantum Computer Studies: Architectural Comparisons and Design Insights. In *Proceedings of the 46th International Symposium on Computer Architecture*, ISCA '19, pages 527–540, New York, NY, USA, 2019. ACM.

[125] 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*, ASPLOS 20, page 10011016, New York, NY, USA, 2020. Association for Computing Machinery.

[126] Y. Nam, N. J. Ross, Y. Su, A. M. Childs, and D. Maslov. Automated optimization of large quantum circuits with continuous parameters. *ArXiv e-prints*, October 2017.

[127] M. Neeley, R. C. Bialczak, M. Lenander, E. Lucero, M. Mariantoni, A. D. O'Connell, D. Sank, H. Wang, M. Weides, J. Wenner, Y. Yin, T. Yamamoto, A. N. Cleland, and J. M. Martinis. Generation of three-qubit entangled states using superconducting phase qubits. *Nature*, 467:570–573, September 2010.

[128] C. Neill, P. Roushan, K. Kechedzhi, S. Boixo, S. V. Isakov, V. Smelyanskiy, A. Megrant, B. Chiaro, A. Dunsworth, K. Arya, R. Barends, B. Burkett, Y. Chen, Z. Chen, A. Fowler, B. Foxen, M. Giustina, R. Graff, E. Jeffrey, T. Huang, J. Kelly, P. Klimov, E. Lucero, J. Mutus, M. Neeley, C. Quintana, D. Sank, A. Vainsencher, J. Wenner, T. C. White, H. Neven, and J. M. Martinis. A blueprint for demonstrating quantum supremacy with superconducting qubits. *Science*, 360:195–199, April 2018.

[129] Luke Nelson, James Bornholt, Ronghui Gu, Andrew Baumann, Emina Torlak, and Xi Wang. Scaling symbolic evaluation for automated verification of systems code with serval. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, SOSP '19, pages 225–242, New York, NY, USA, 2019. ACM.

[130] Luke Nelson, Helgi Sigurbjarnarson, Kaiyuan Zhang, Dylan Johnson, James Bornholt, Emina Torlak, and Xi Wang. Hyperkernel: Push-button verification of an os kernel. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 252–269, New York, NY, USA, 2017. ACM.

[131] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, New York, NY, USA, 10th edition, 2011.

[132] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, New York, NY, USA, 10th edition, 2011.

[133] R. Nigmatullin and S. G. Schirmer. Implementation of fault-tolerant quantum logic gates via optimal control. *New Journal of Physics*, 11(10):105032, October 2009.

[134] K Noh, V V Albert, and L Jiang. Quantum Capacity Bounds of Gaussian Thermal Loss Channels and Achievable Rates With Gottesman-Kitaev-Preskill Codes. *IEEE Transactions on Information Theory*, 65(4):2563–2582, apr 2019.

[135] Kyungjoo Noh and Christopher Chamberland. Fault-tolerant bosonic quantum error correction with the surface-GKP code. *arXiv e-prints*, page arXiv:1908.03579, Aug 2019.

[136] Nissim Ofek, Andrei Petrenko, Reinier Heeres, Philip Reinhold, Zaki Leghtas, Brian Vlastakis, Yehan Liu, Luigi Frunzio, S. M. Girvin, Liang Jiang, Mazyar Mirrahimi, M. H. Devoret, and R. J. Schoelkopf. Demonstrating quantum error correction that extends the lifetime of quantum information. *arXiv:quant-ph/1602.04768*, 2016.

[137] Yukiyoshi Ohtsuki, Yoshiaki Teranishi, Peter Saalfrank, Gabriel Turinici, and Herschel Rabitz. Monotonically convergent algorithms for solving quantum optimal control problems described by an integrodifferential equation of motion. *Phys. Rev. A*, 75:033407, March 2007.

[138] Yukiyoshi Ohtsuki, Gabriel Turinici, and Herschel Rabitz. Generalized monotonically convergent algorithms for solving quantum optimal control problems. *The Journal of Chemical Physics*, 120(12):5509–5517, March 2004.

[139] T. P. Orlando, J. E. Mooij, Lin Tian, Caspar H. van der Wal, L. S. Levitov, Seth Lloyd, and J. J. Mazo. Superconducting persistent-current qubit. *Phys. Rev. B*, 60:15398–15413, Dec 1999.

[140] H. Paik, A. Mezzacapo, M. Sandberg, D. T. McClure, B. Abdo, A. D. Córcoles, O. Dial, D. F. Bogorin, B. L. T. Plourde, M. Steffen, A. W. Cross, J. M. Gambetta, and J. M. Chow. Experimental Demonstration of a Resonator-Induced Phase Gate in a Multiqubit Circuit-QED System. *Physical Review Letters*, 117(25):250502, December 2016.

[141] José P. Palao and Ronnie Kosloff. Optimal control theory for unitary transformations. *Phys. Rev. A*, 68(6):062308+, December 2003.

[142] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Aln Aspuru-Guzik, and Jeremy L. OBrien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1), Jul 2014.

[143] John Preskill. Quantum Computing in the NISQ era and beyond. *ArXiv e-prints*, jan 2018.

[144] Shruti Puri, Alexander Grimm, Philippe Campagne-Ibarcq, Alec Eickbusch, Kyungjoo Noh, Gabrielle Roberts, Liang Jiang, Mazyar Mirrahimi, Michel H. Devoret, and Steven M. Girvin. Stabilized Cat in Driven Nonlinear Cavity: A Fault-Tolerant Error Syndrome Detector. *arXiv e-prints*, page arXiv:1807.09334, Jul 2018.

[145] IBM Qiskit. Github: qiskit-backend-information. https://github.com/Qiskit/qiskit-backend-information, July 2018.

[146] Qiskit bug report, 2018.

[147] Robert Rand, Jennifer Paykin, Dong-Ho Lee, and Steve Zdancewic. ReQWIRE: Reasoning about Reversible Quantum Circuits. *arXiv e-prints*, page arXiv:1901.10118, Jan 2019.

[148] Robert Rand, Jennifer Paykin, and Steve Zdancewic. QWIRE Practice: Formal Verification of Quantum Circuits in Coq. *Electronic Proceedings in Theoretical Computer Science*, 2018.

[149] Matthew Reagor, Hanhee Paik, Gianluigi Catelani, Luyan Sun, Christopher Axline, Eric Holland, Ioan M. Pop, Nicholas A. Masluk, Teresa Brecht, Luigi Frunzio, Michel H. Devoret, Leonid Glazman, and Robert J. Schoelkopf. Reaching 10 ms single photon lifetimes for superconducting aluminum cavities. *Applied Physics Letters*, 102(19):192604, 2013.

[150] Daniel M. Reich, Mamadou Ndong, and Christiane P. Koch. Monotonically convergent optimization in quantum control using Krotov's method. *The Journal of Chemical Physics*, 136(10):104103, March 2012.

[151] Ben W. Reichardt. Fault-tolerant quantum error correction for steane's seven-qubit color code with few or no extra qubits. *arXiv:quant-ph/1804.06995*, 2018.

[152] J. Romero, R. Babbush, J. R. McClean, C. Hempel, P. Love, and A. Aspuru-Guzik. Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz. *ArXiv e-prints*, January 2017.

[153] S. Rosenblum, P. Reinhold, M. Mirrahimi, Liang Jiang, L. Frunzio, and R. J. Schoelkopf. Fault-tolerant detection of a quantum error. *Science*, 361(6399):266–270, 2018.

[154] Neil J. Ross and Peter Selinger. Optimal ancilla-free clifford+t approximation of z-rotations, 2014.

[155] N. Schuch and J. Siewert. Natural two-qubit gate for quantum computation using the XY interaction. *Physical Review A*, 67(3):032301, March 2003.

[156] T. Schulte-Herbrueggen, A. Spoerl, and S. J. Glaser. Quantum CISC Compilation by Optimal Control and Scalable Assembly of Complex Instruction Sets beyond Two-Qubit Gates. *ArXiv e-prints*, December 2007.

[157] Peter Selinger. Efficient clifford+t approximation of single-qubit operators, 2012.

[158] Kanav Setia, Sergey Bravyi, Antonio Mezzacapo, and James D. Whitfield. Superfast encodings for fermionic quantum simulation, 2018.

[159] 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*, ASPLOS '19, pages 1031–1044, New York, NY, USA, 2019. ACM. http://doi.acm.org/10.1145/3297858.3304018.

[160] 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 *ASPLOS*, Providence, Rhode Island, 2019.

[161] Peter W. Shor. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 52:2493, 1995.

[162] Peter W. Shor. Fault-tolerant quantum computation. *Proceedings., 37th Annual Symposium on Foundations of Computer Science*, pages 56–65, 1996.

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

[164] P.W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. *Proceedings., 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[165] Helgi Sigurbjarnarson, James Bornholt, Emina Torlak, and Xi Wang. Push-Button Verification of File Systems via Crash Refinement. In *the Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI*, 2016.

[166] Kartik Singhal, Robert Rand, and Michael Hicks. Verified translation between low-level quantum languages. The First International Workshop on Programming Languages for Quantum Computing, jan 2020.

[167] Shlomo E. Sklarz and David J. Tannor. Loading a Bose-Einstein condensate onto an optical lattice: An application of optimal control theory to the nonlinear Schrödinger equation. *Phys. Rev. A*, 66(5):053619, November 2002.

[168] R. S. Smith, M. J. Curtis, and W. J. Zeng. A Practical Quantum Instruction Set Architecture. *ArXiv e-prints*, August 2016.

[169] Robert S Smith, Michael J Curtis, and William J Zeng. A practical quantum instruction set architecture, 2016.

[170] Mathias Soeken and Michael Kirkedal Thomsen. White dots do matter: Rewriting reversible logic circuits. In Gerhard W. Dueck and D. Michael Miller, editors, *Reversible Computation*, pages 196–208, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[171] Andrew W. Steane. Error correcting codes in quantum theory. *Phys. Rev. Lett.*, 77:793, 1996.

[172] Strawberry fields, 2016.

[173] Wang-Chang Su, Chai-Yu Lin, and Shin-Tza Wu. Encoding qubits into harmonic-oscillator modes via quantum walks in phase space. *arXiv e-prints*, page arXiv:1808.08722, Aug 2018.

[174] Krysta M. Svore, Matthew B. Hastings, and Michael Freedman. Faster phase estimation. *Quantum Info. Comput.*, 14(3-4):306–328, March 2014.

[175] DavidJ Tannor, Vladimir Kazakov, and Vladimir Orlov. Control of Photochemical Branching: Novel Procedures for Finding Optimal Pulses and Global Upper Bounds. In J. Broeckhove and L. Lathouwers, editors, *Time-Dependent Quantum Molecular Dynamics*, volume 299 of *Nato ASI Series*, pages 347–360. Springer US, 1992.

[176] 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*, MICRO '52, pages 253–265, New York, NY, USA, 2019. ACM.

[177] 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*, ASPLOS 19, page 987999, New York, NY, USA, 2019. Association for Computing Machinery.

[178] Theerapat Tansuwannont, Christopher Chamberland, and Debbie Leung. Flag fault-tolerant error correction for cyclic css codes. *arXiv:1803.09758*, 2018.

[179] B. M. Terhal and D. Weigand. Encoding a qubit into a cavity mode in circuit qed using phase estimation. *Phys. Rev. A*, 93:012315, Jan 2016.

[180] B. M. Terhal and D. Weigand. Encoding a qubit into a cavity mode in circuit qed using phase estimation. *Phys. Rev. A*, 93:012315, Jan 2016.

[181] Qiskit Terra Github issue page, 2018.

[182] **Yunong Shi**, Christopher Chamberland, and Andrew Cross. Fault-tolerant preparation of approximate GKP states. *New Journal of Physics*, 21(9):093007, sep 2019.

[183] **Yunong Shi**, Pranav Gokhale, Prakash Murali, Jonathan M. Baker, Casey Duckering, Yongshan Ding, Natalie C. Brown, Christopher Chamberland, Ali Javadi Abhari, Andrew W. Cross, David I. Schuster, Kenneth R. Brown, Margaret R. Martonosi, and Frederic T. Chong. Resource-efficient quantum computingby breaking abstractions. *PIEEE Special Issue*, Jan 2020.

[184] **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*, ASPLOS '19, pages 1031–1044, New York, NY, USA, 2019. ACM.

[185] **Yunong Shi**, Xupeng Li, Runzhou Tao, Ali Javadi-Abhari, Andrew W. Cross, Frederic T. Chong, and Ronghui Gu. Contract-based verification of a realistic quantum compiler. *arXiv e-prints*, Aug 2019.

[186] B. C. Travaglione and G. J. Milburn. Preparing encoded states in an oscillator. *Phys. Rev. A*, 66:052322, Nov 2002.

[187] B. C. Travaglione and G. J. Milburn. Preparing encoded states in an oscillator. *Phys. Rev. A*, 66:052322, Nov 2002.

[188] L. M. K. Vandersypen and I. L. Chuang. NMR techniques for quantum control and computation. *Reviews of Modern Physics*, 76:1037–1069, October 2004.

[189] George F. Viamontes, Igor L. Markov, and John P. Hayes. Checking equivalence of quantum circuits and states. In Georges G E Gielen, editor, *ICCAD*. IEEE, 2007.

[190] Christophe Vuillot, Hamed Asasi, Yang Wang, Leonid P. Pryadko, and Barbara M. Terhal. Quantum error correction with the toric gottesman-kitaev-preskill code. *Phys. Rev. A*, 99:032344, Mar 2019.

[191] Joel J. Wallman and Joseph Emerson. Noise tailoring for scalable quantum computation via randomized compiling. *Physical Review A*, 94(5), Nov 2016.

[192] Christopher J. Wood and Jay M. Gambetta. Quantification and characterization of leakage errors. *Physical Review A*, 97(3), mar 2018.

[193] Mingsheng Ying. Floyd–hoare logic for quantum programs. *ACM Transactions on Programming Languages and Systems*, 33(6):1–49, dec 2011.

[194] Wusheng Zhu, Jair Botina, and Herschel Rabitz. Rapidly convergent iteration methods for quantum optimal control of population. *The Journal of Chemical Physics*, 108(5):1953–1963, February 1998.

[195] Wusheng Zhu and Herschel Rabitz. A rapid monotonically convergent iteration algorithm for quantum optimal control over the expectation value of a positive definite operator. *The Journal of Chemical Physics*, 109(2):385–391, July 1998.

[196] Alwin Zulehner, Alexandru Paler, and Robert Wille. An efficient methodology for mapping quantum circuits to the ibm qx architectures, 2017.