Partial Compilation of Variational Algorithms for Noisy Intermediate-Scale Quantum Machines

Pranav Gokhale[1] Yongshan Ding[1] Thomas Propson[2]
Christopher Winkler[1] Nelson Yeung[2] Yunong Shi[2]
David I. Schuster[2] Henry Hoffmann[1] Frederic T. Chong[1]

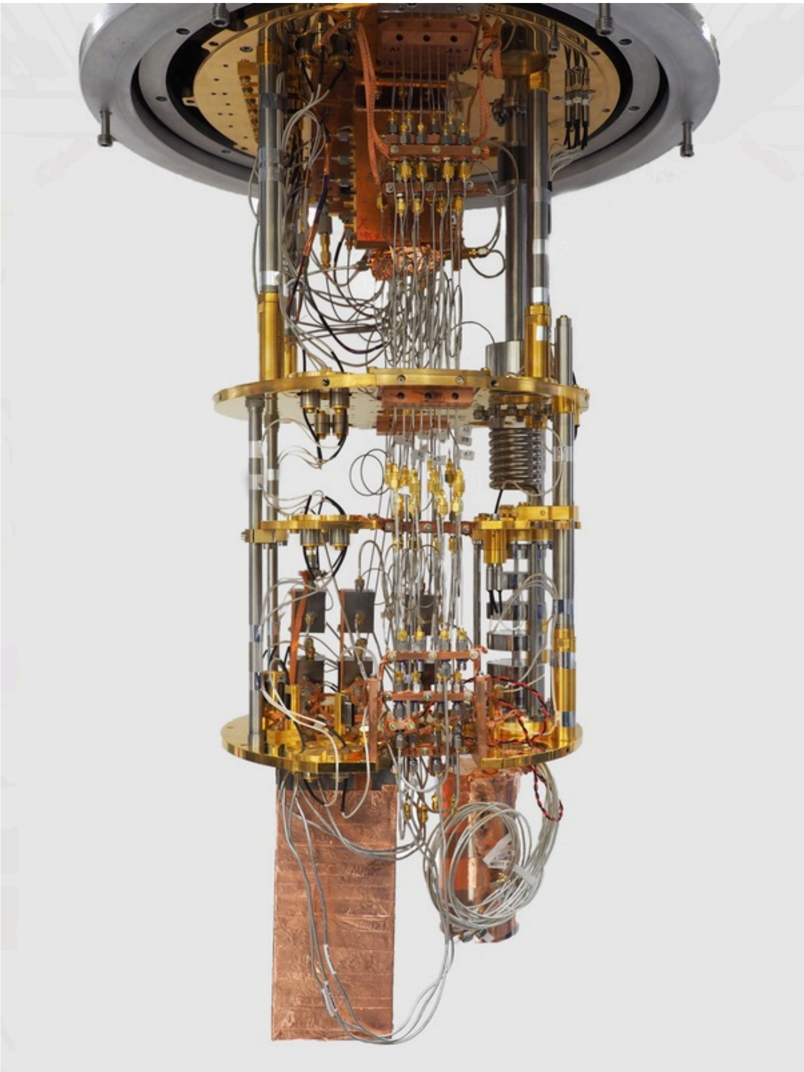
[1]Department of Computer Science
University of Chicago

[2]Department of Physics
University of Chicago

MICRO-52, October 2019

Outline

- Background: Quantum Compilation
- Variational Quantum Algorithms
- Partial Compilation
  - Strict
  - Flexible
- Results
- Conclusions / Future Work



Background

arXiv:1902.01474v2 [quant-ph] 17 Feb 2019

Optimized Compilation of Aggregated Instructions for Realistic Quantum Computers

Yunong Shi
The University of Chicago
yunong@uchicago.edu

Nelson Leung
The University of Chicago
nelsonleung@uchicago.edu

Pranav Gokhale
The University of Chicago
pranavgokhale@uchicago.edu

Zane Rossi
The University of Chicago
zmr@uchicago.edu

David I. Schuster
The University of Chicago
david.schuster@uchicago.edu

Henry Hoffmann
The University of Chicago
hankhoffmann@cs.uchicago.edu

Frederic T. Chong
The University of Chicago
chong@cs.uchicago.edu

Abstract

Recent developments in engineering and algorithms have made real-world applications in quantum computing possible in the near future. Existing quantum programming languages and compilers use a quantum assembly language composed of 1- and 2-qubit (quantum bit) gates. Quantum compiler frameworks translate this quantum assembly to electric signals (called control pulses) that implement the specified computation on specific physical devices. However, there is a mismatch between the operations defined by the 1- and 2-qubit logical ISA and their underlying physical implementation, so the current practice of directly translating logical instructions into control pulses results in inefficient, high-latency programs. To address this inefficiency, we propose a universal quantum compilation methodology that aggregates multiple logical operations into larger units that manipulate up to 10 qubits at a time. Our methodology then optimizes these aggregates by (1) finding commutative intermediate operations that result in more efficient schedules and (2) creating custom control pulses optimized for the aggregate (instead of individual 1- and 2-qubit operations). Compared to the standard gate-based compilation, the proposed approach realizes a deeper vertical integration of high-level quantum software and low-level, physical quantum hardware. We evaluate our approach on important near-term quantum applications on simulations of superconducting quantum architectures. Our proposed approach provides a mean speedup of 5×, with a maximum of 10×. Because latency directly affects the feasibility of quantum computation, our results not only improve performance but also have the potential to enable quantum computation sooner than otherwise possible.

ACM Reference Format:
Yunong Shi, Nelson Leung, Pranav Gokhale, Zane Rossi, David I. Schuster, Henry Hoffmann, and Frederic T. Chong. 2019. Optimized Compilation of Aggregated Instructions for Realistic Quantum Computers. In *2019 Architectural Support for Programming Languages and Operating Systems (ASPLOS '19), April 13–17, 2019, Providence, RI, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3297858.3304018

1 Introduction

The past twenty years have seen the world of quantum computing moving closer to solving classically intractable problems [2, 9, 50]. With developments in Noisy Intermediate-Scale Quantum (NISQ) [45] devices like IBM's quantum machine with 50 qubits and Google's quantum machine with 72 qubits, we may soon be able to demonstrate computations not possible on classical supercomputers [2, 9]. Exciting classical-quantum hybrid algorithms tailored for NISQ machines, like Quantum Approximate Optimization Algorithm (QAOA) [8] and Variational Quantum Eigensolver (VQE) [36, 44] will power up the first real-world quantum computing applications with scientific and commercial value.

Computation latency is a major challenge for near-term quantum computing. While all computing systems benefit from reduced latency, in a quantum system the output fidelity decays at least exponentially with latency [41]. Thus, in near-term quantum computers, reducing latency is not just a minor convenience—latency reduction actually enables new computations on near-term machines by ensuring that the computation finishes before the qubits decohere and produce a useless result. Thus latency reduction is critical

ASPLOS '19, April 13–17, 2019, Providence, RI, USA

ACM ISBN 978-1-4503-6240-5/19/04...$15.00
https://doi.org/10.1145/3297858.3304018

ASPLOS 2019
arXiv:1902.01474

- 5x improvement in quantum runtime
- Qubits decay **exponentially** with time
- How? Compilation to pulses

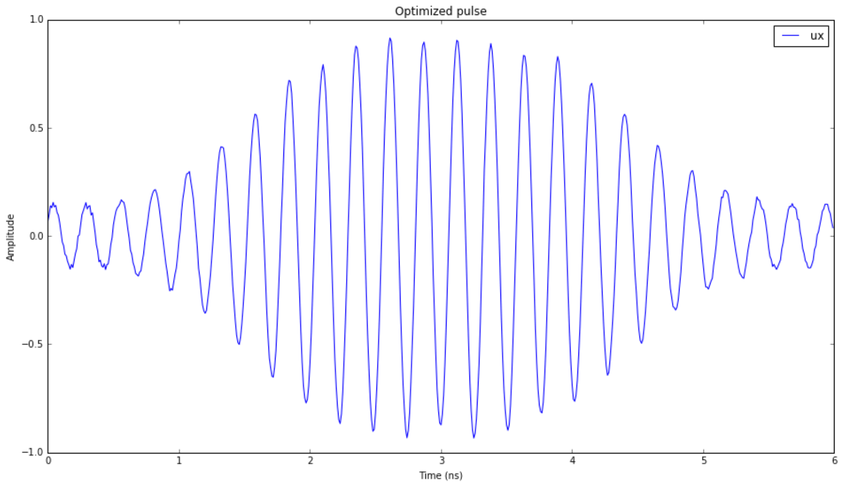
Background

```
module QFT(qbit x[]) {
  int i, j;
  double angle = 0.0;
  for (i=0; i<_n; i++){
    H(x[i]);
    for (j=i+1; j<_n; j++) {
      angle = angle_R[j];
      cRz(x[j], x[i], angle);
```

Programming Language

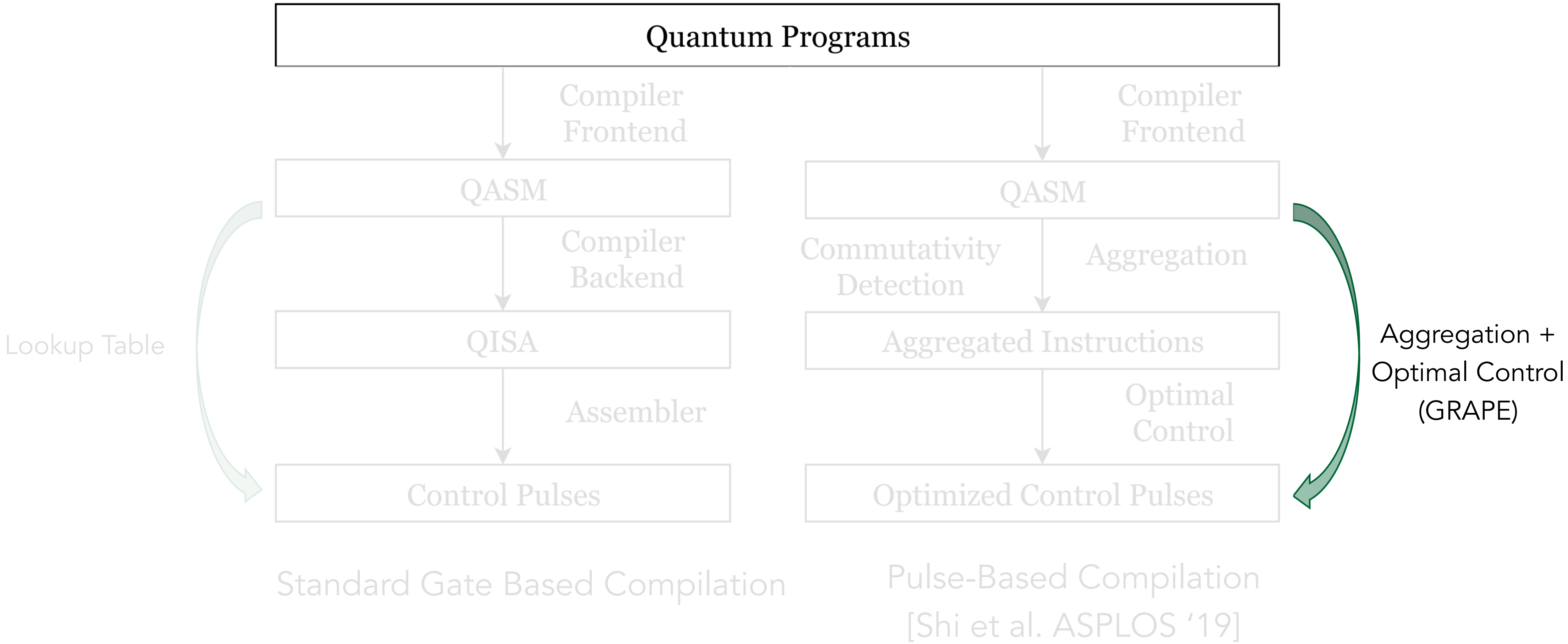
```
x q[2];
barrier q;
h q[0];
cu1(pi/2) q[1],q[0];
h q[1];
cu1(pi/4) q[2],q[0];
cu1(pi/2) q[2],q[1];
h q[2];
cu1(pi/8) q[3],q[0];
cu1(pi/4) q[3],q[1];
cu1(pi/2) q[3],q[2];
h q[3];
measure q -> c;
```

QASM

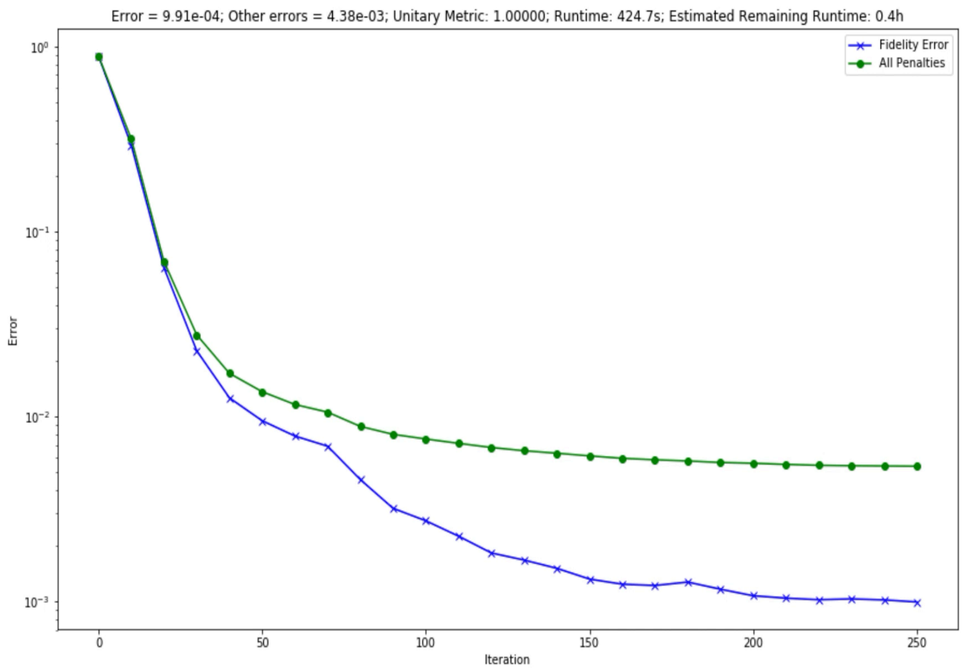


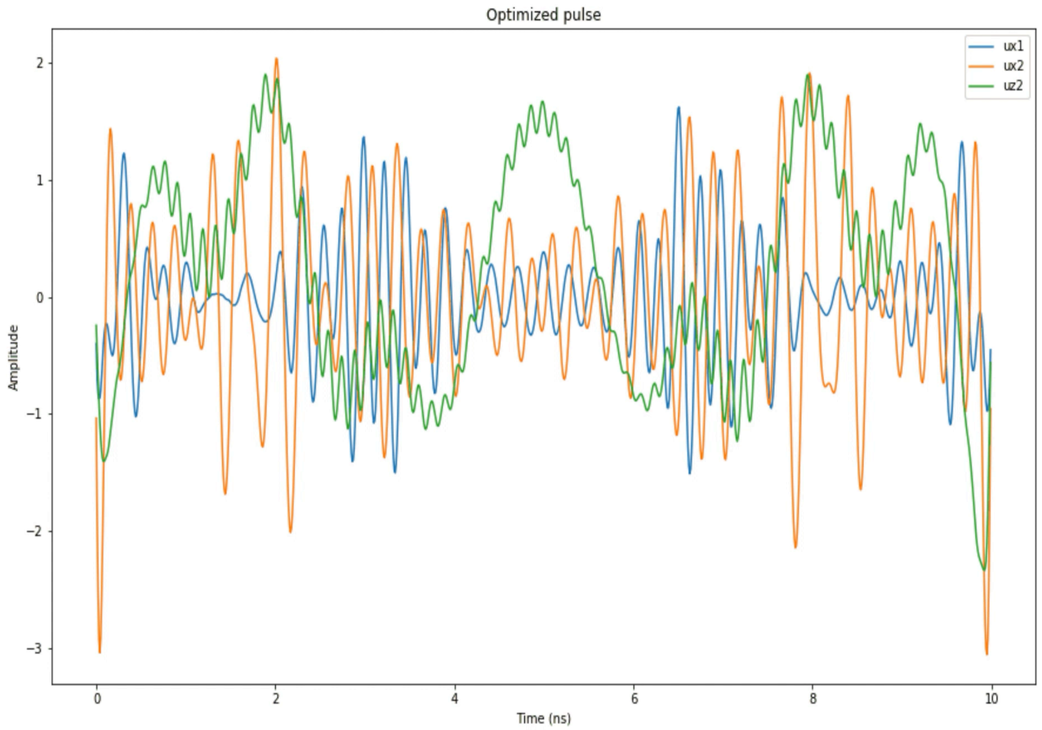
Pulses

Background

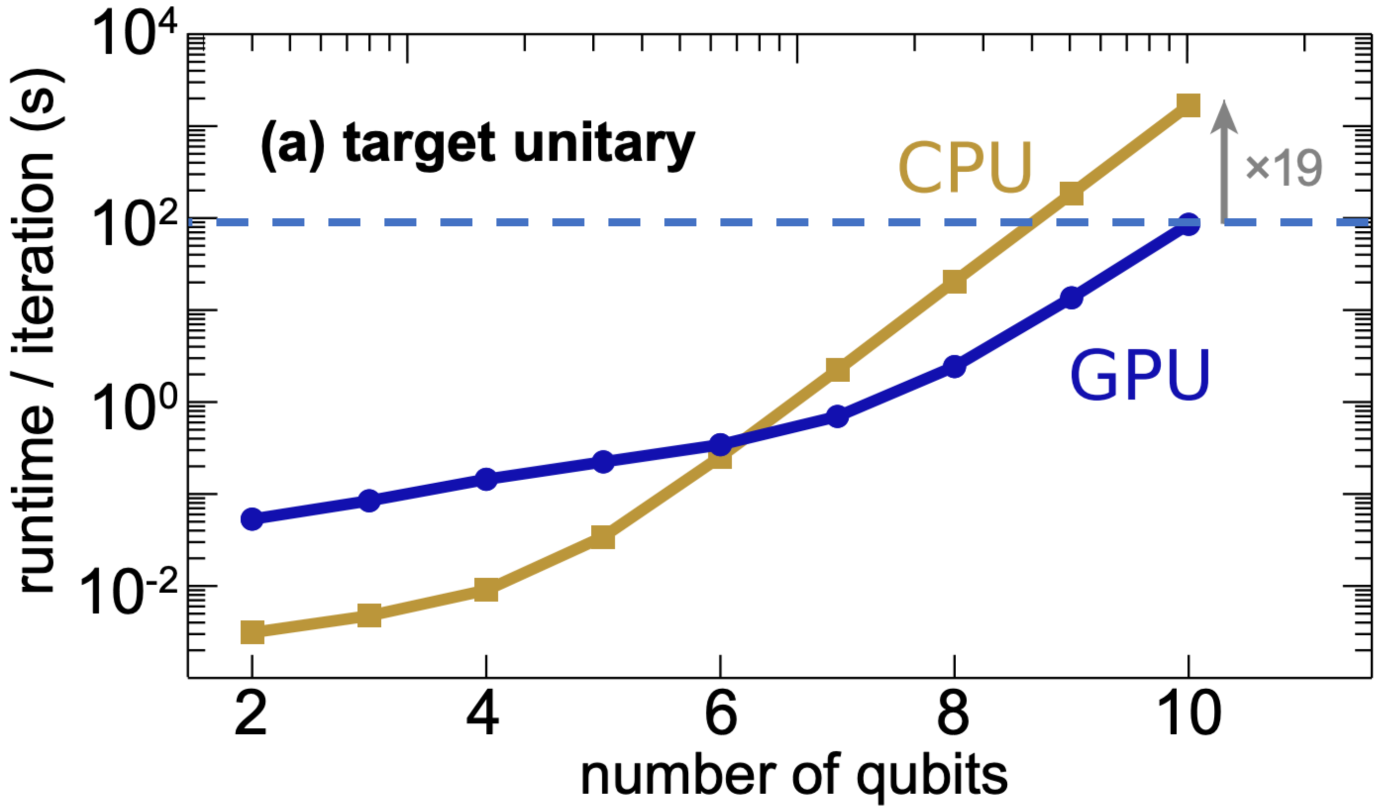

Quantum Programs
Compiler Frontend
QASM
Compiler Backend
QISA
Assembler
Control Pulses
Lookup Table
Compiler Frontend
QASM
Commutativity Detection
Aggregation
Aggregated Instructions
Optimal Control
Optimized Control Pulses
Aggregation + Optimal Control (GRAPE)
Standard Gate Based Compilation
Pulse-Based Compilation [Shi et al. ASPLOS '19]

GRAPE (GRAdient-ascent Pulse Engineering)





GRAPE (GRAdient-ascent Pulse Engineering)



[Leung et al. 2017]

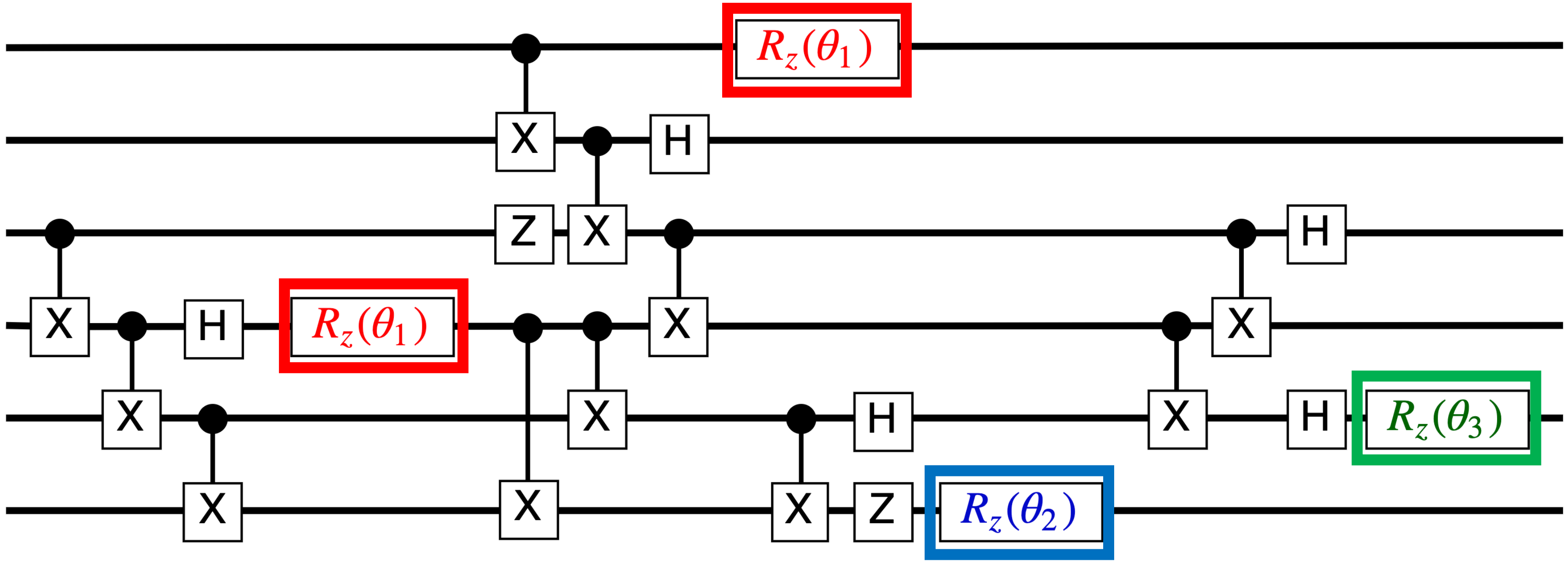
In sum: ASPLOS paper demonstrated how to get large pulse speedups, but takes a long time to compile.

Variational Quantum Algorithms

- Originally quantum algorithms are fully specified at compile time
  - Shor Factoring, Grover Search, etc.
- But, hardware requirements are untenable for Noisy Intermediate-Scale Quantum (NISQ) devices
- Variational quantum algorithms match NISQ hardware

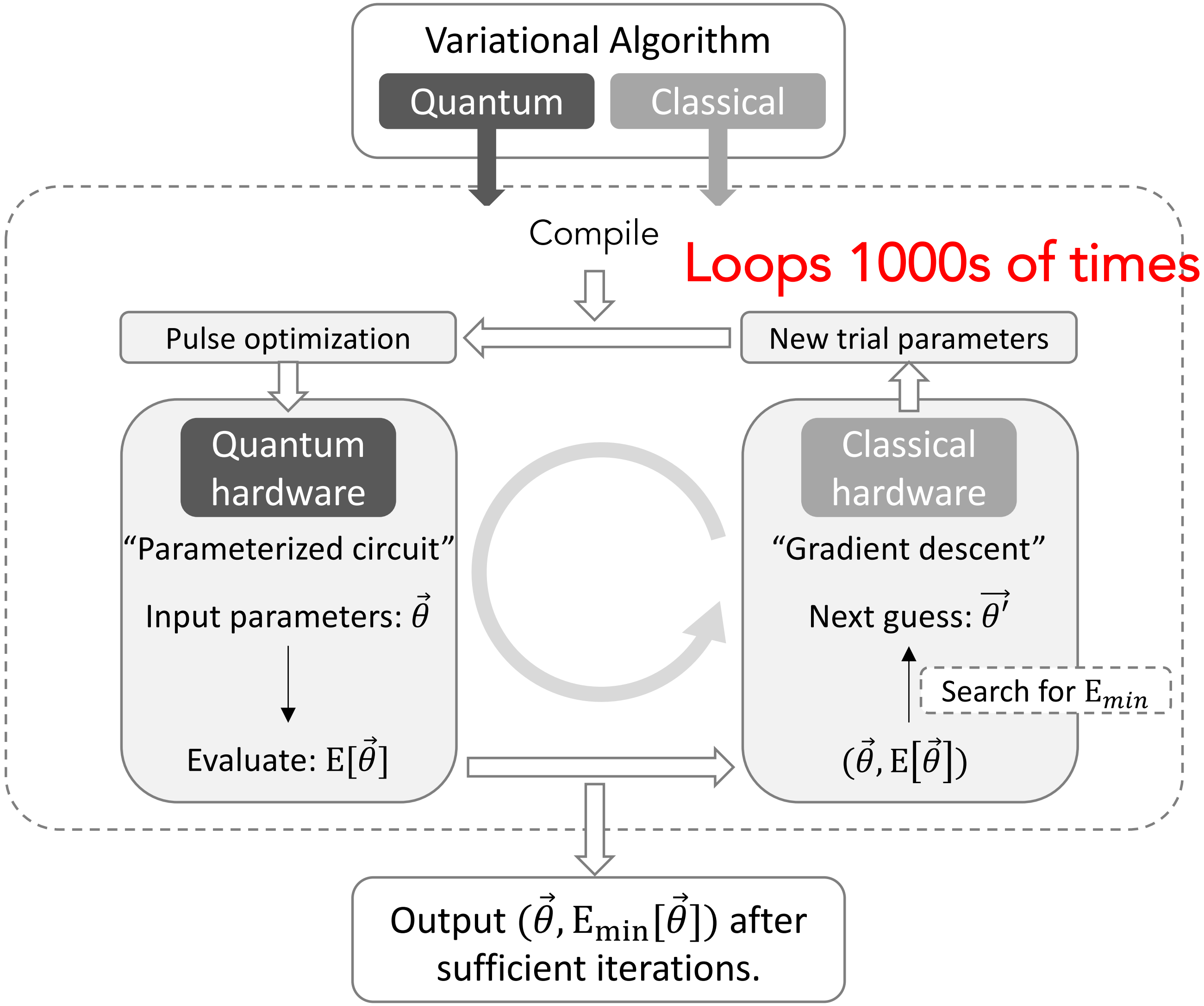
Variational Quantum Algorithms

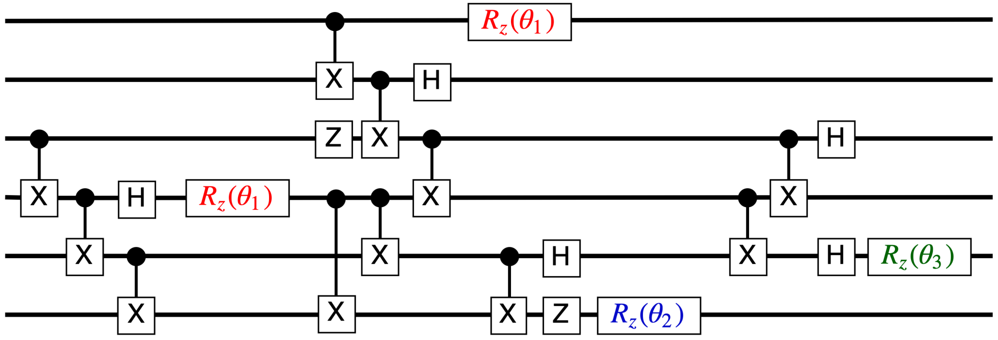
- Quantum program is not fully specified at compile time



- Program is executed multiple times, varying choices of unspecified parameters
- Classical co-processor optimizes choice of parameters, based on history of past executions

Variational Quantum Algorithms





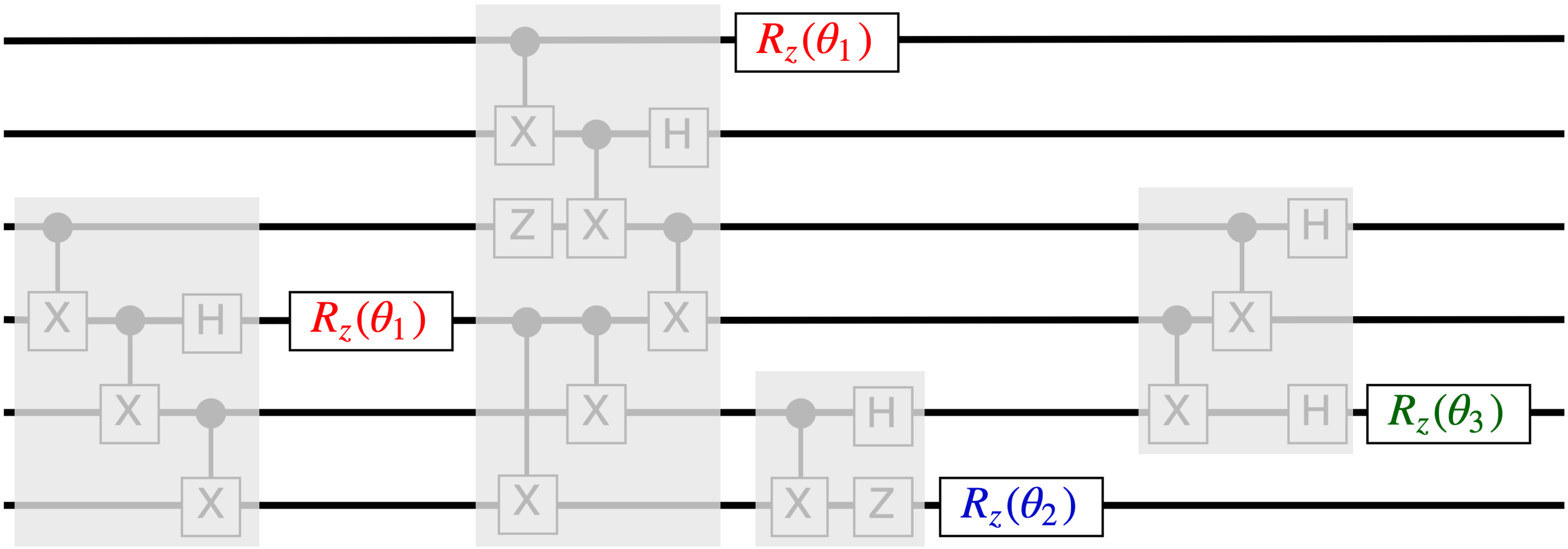
Variational Quantum Algorithms

- New challenge for pulse-level compilation
- Need to compile thousands of programs
- And compilation latency delays time-to-solution

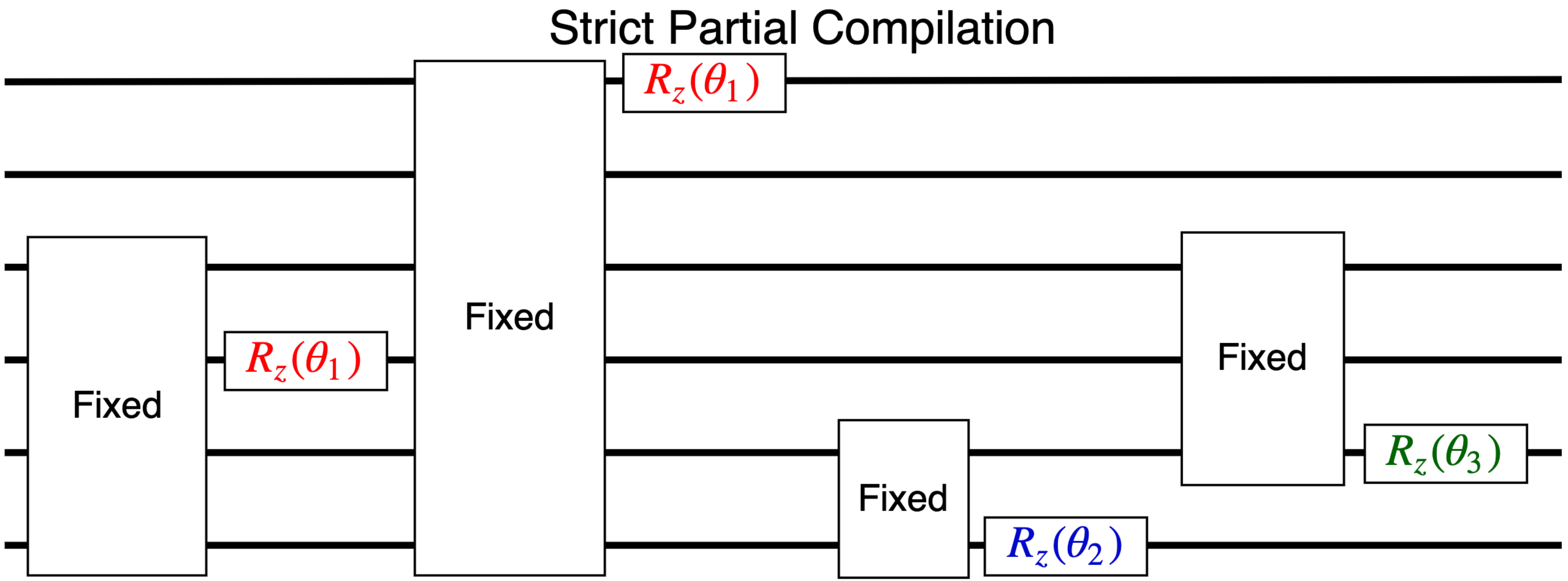
Partial Compilation

- Natural response to a partially specified program
- Two flavors:
  - Strict partial compilation. Pre-compile parts of the program.
  - Flexible partial compilation. Pre-*compute* good hyperparameters to speed up future compilation.

Strict Partial Compilation



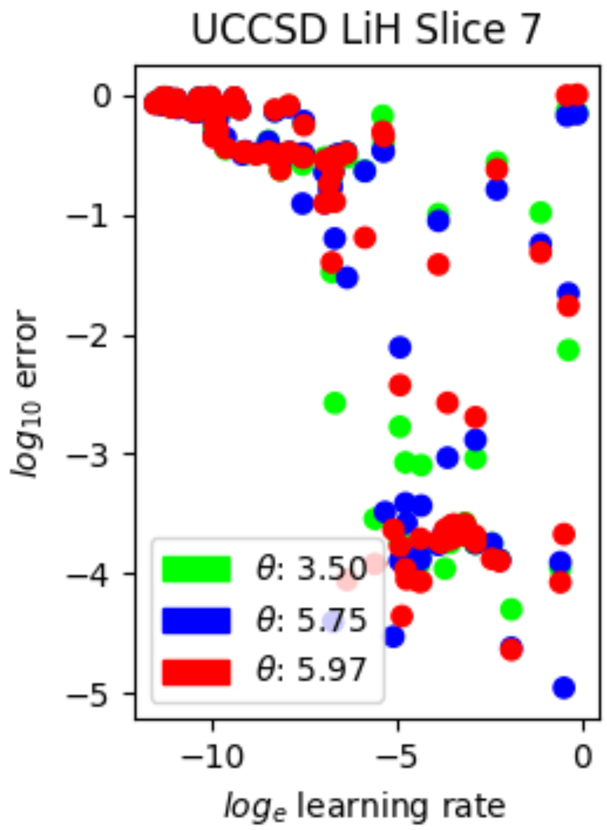
Strict Partial Compilation



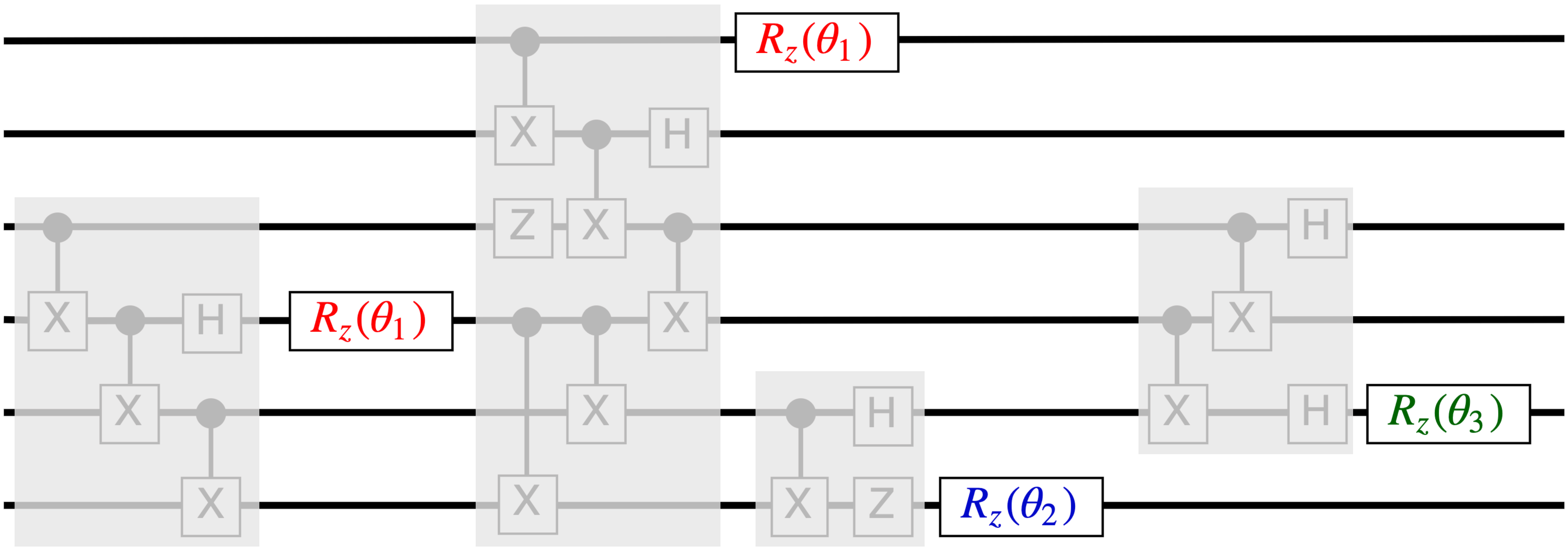
Strict partial compilation *pre-compiles* Fixed blocks—no runtime latency.

Flexible Partial Compilation

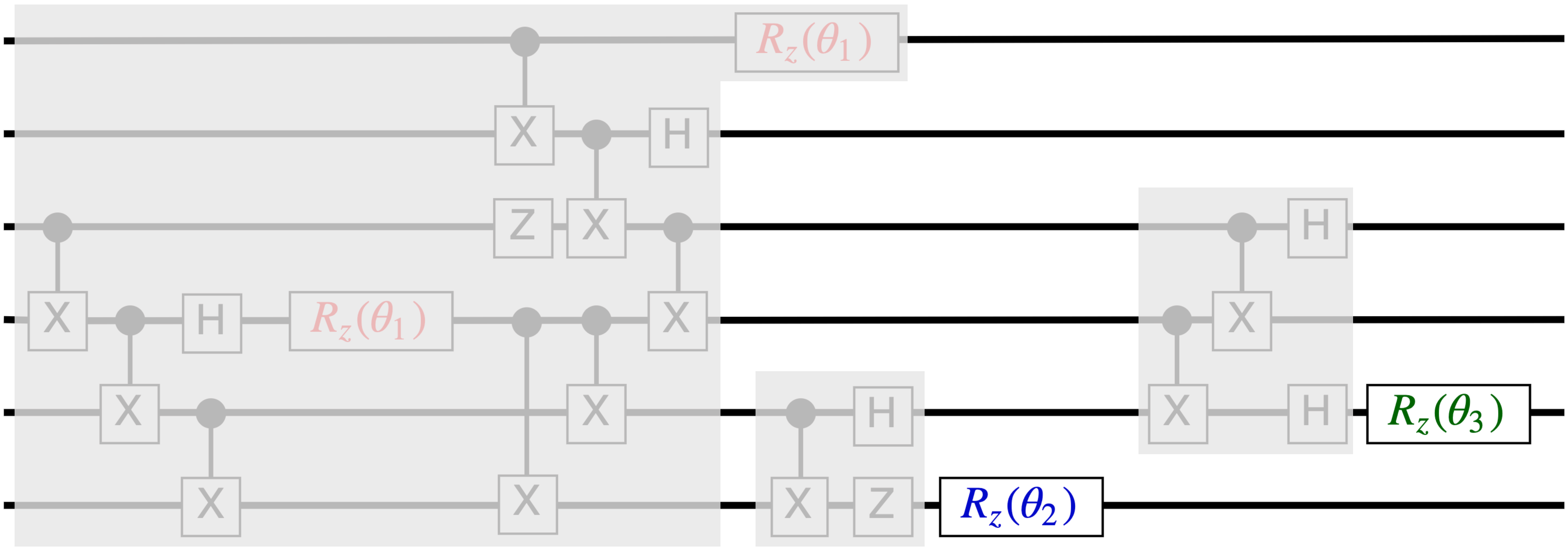
- GRAPE does best with deep (wide) slices
- Strict is limited by interspersed parametrization-dependent gates
- Instead, consider slices parametrized by exactly 1 parameter
- Can't pre-compile, but …
- Can find good hyperparameters



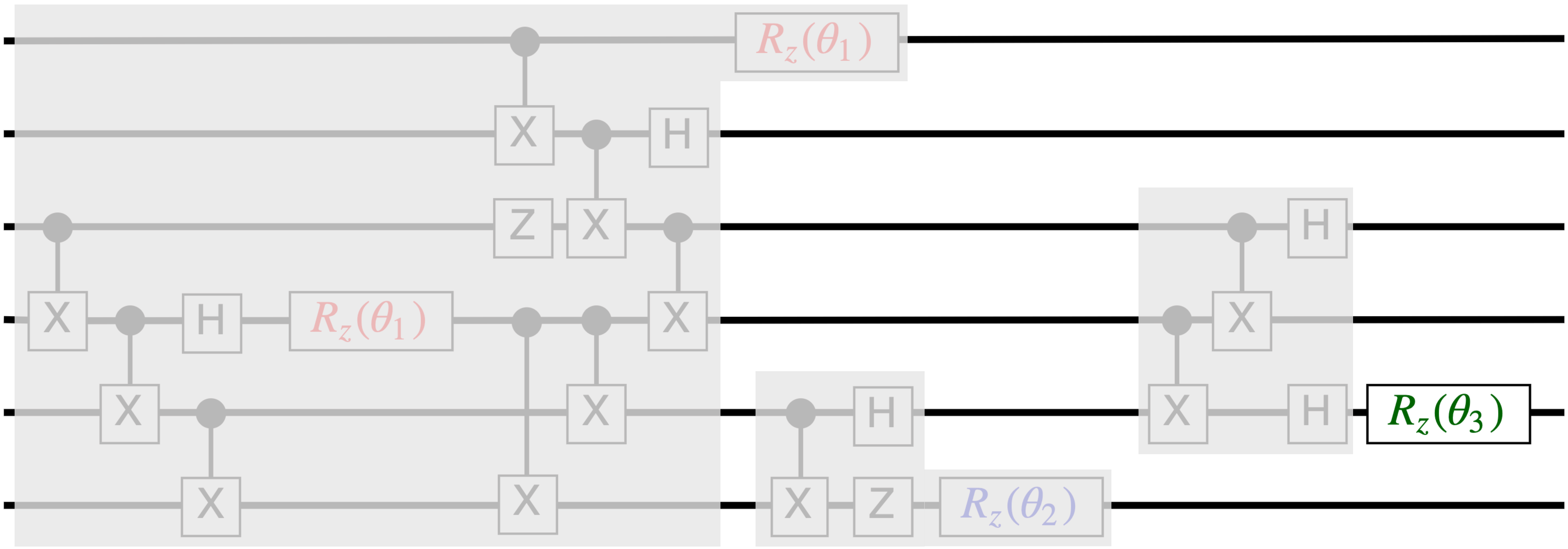
Flexible Partial Compilation



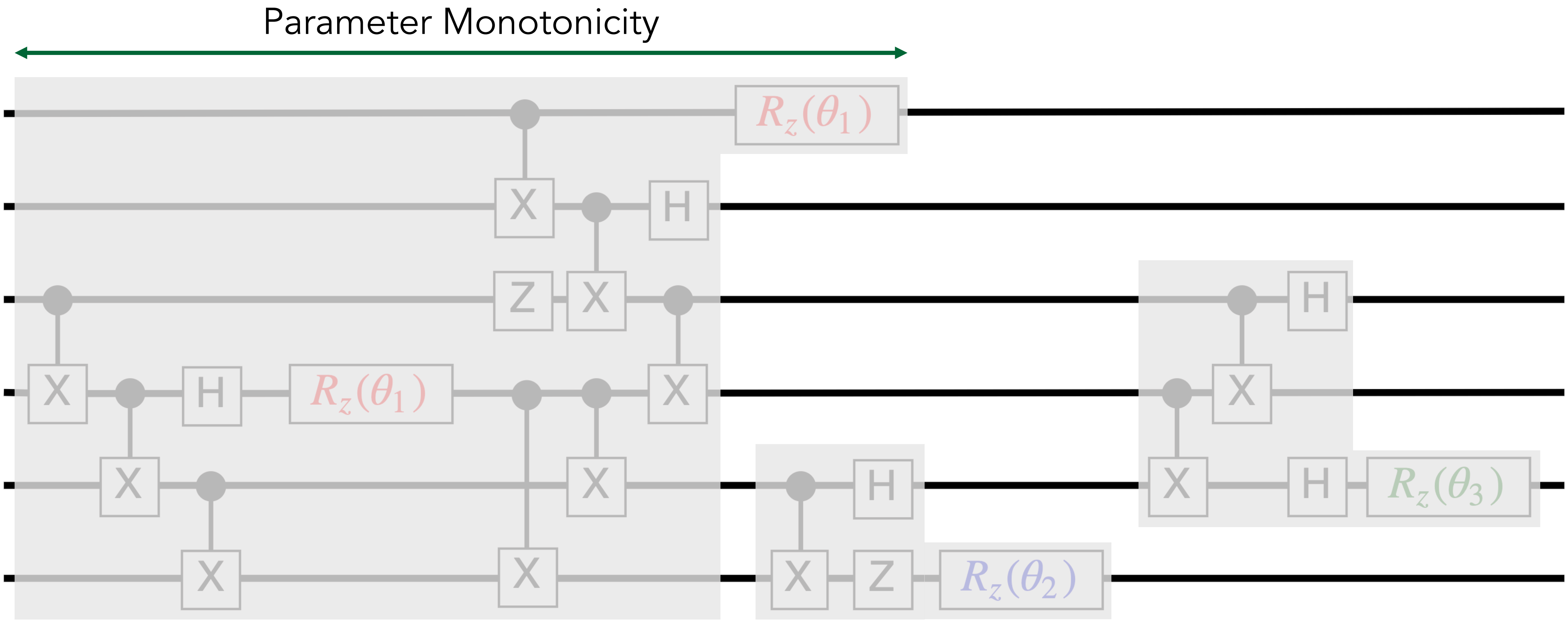
Flexible Partial Compilation



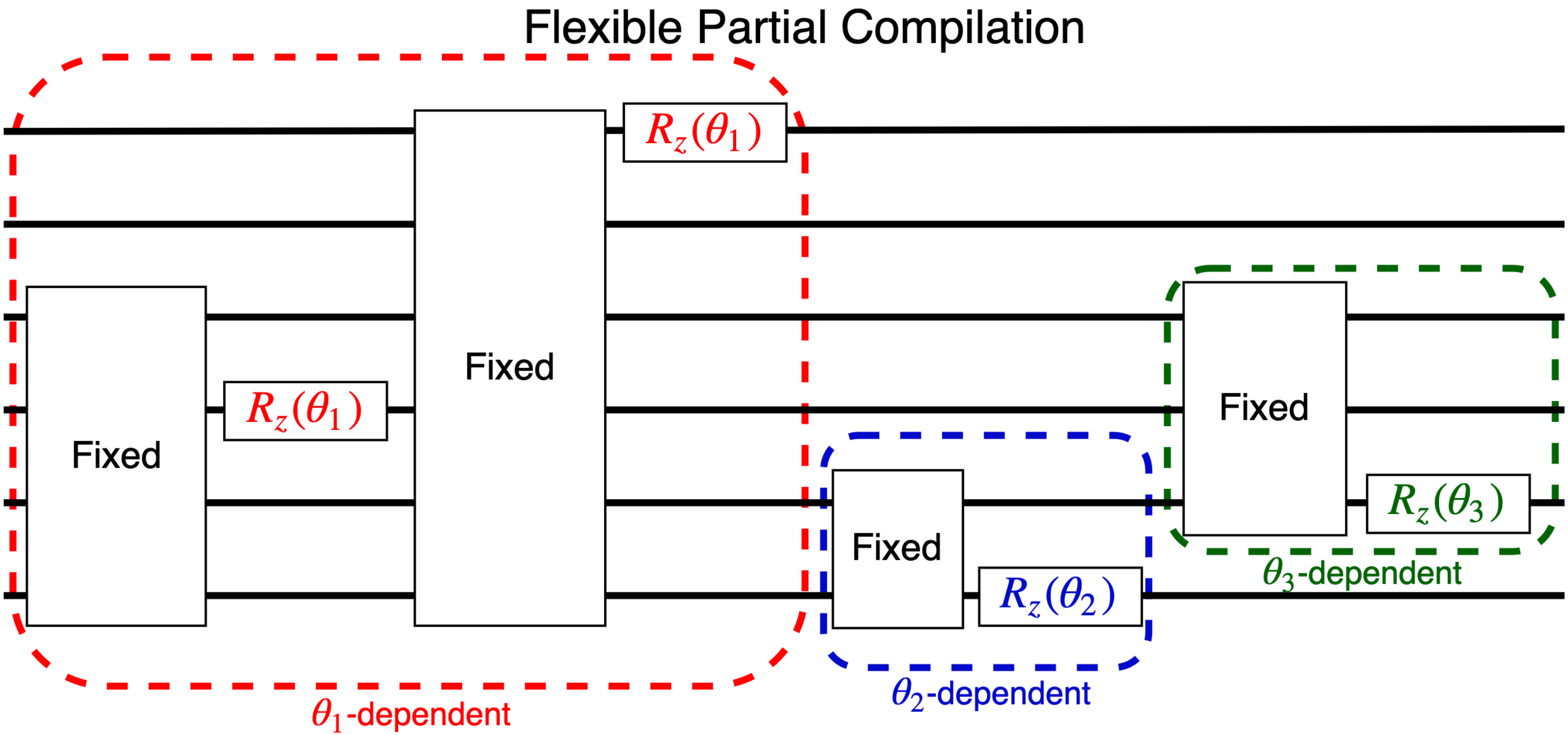
Flexible Partial Compilation



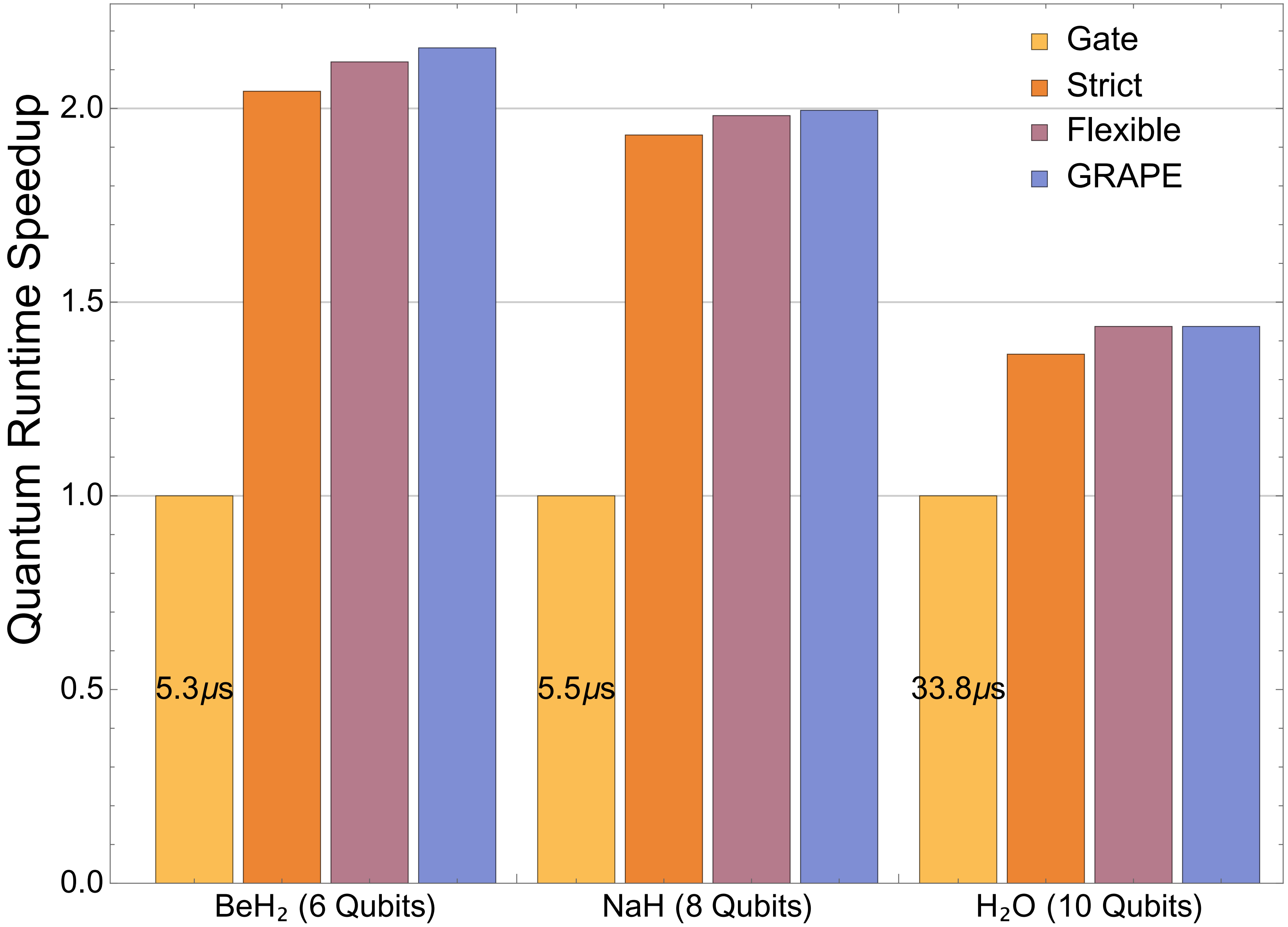
Flexible Partial Compilation



Flexible Partial Compilation

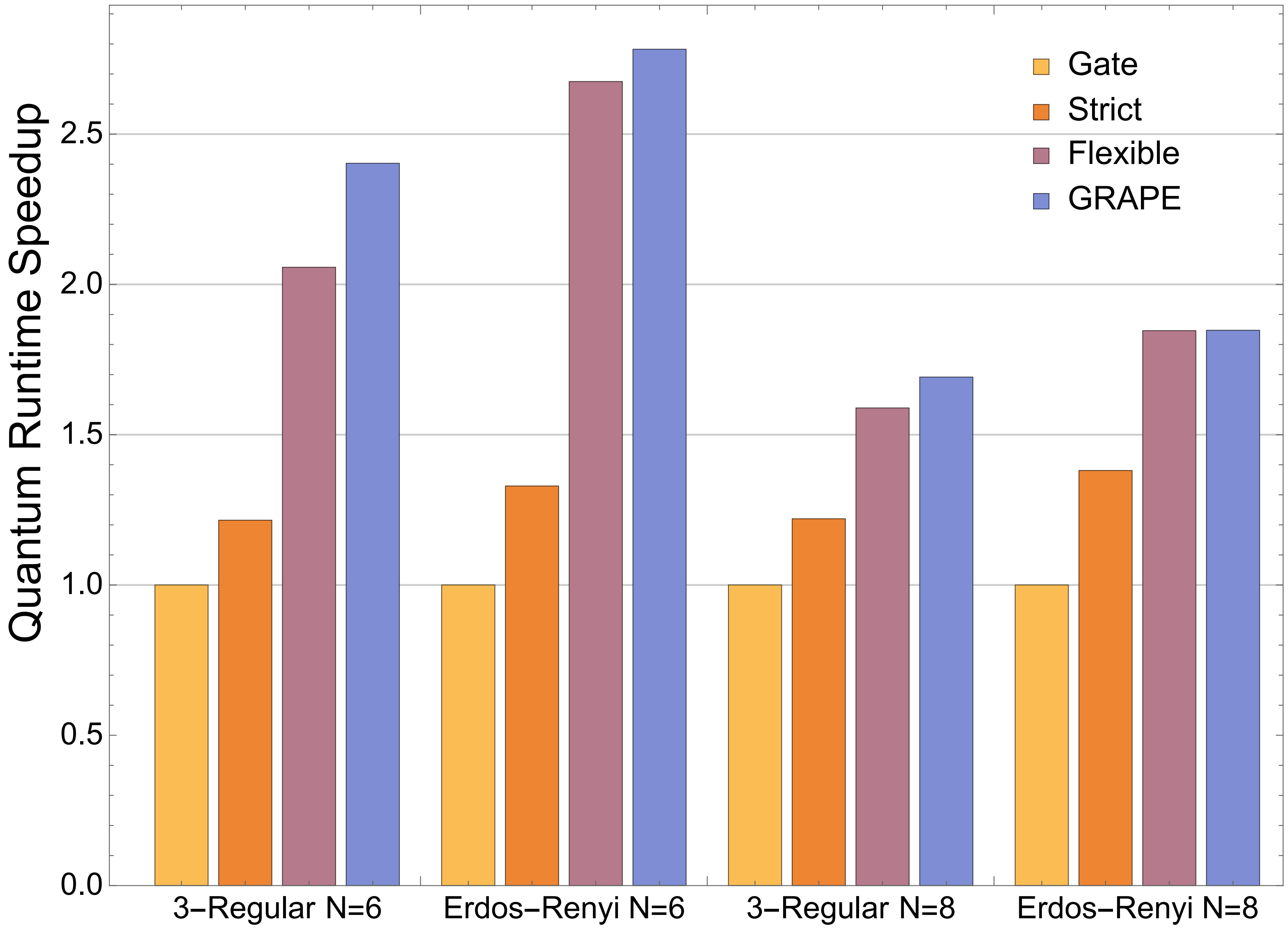


Results: VQE

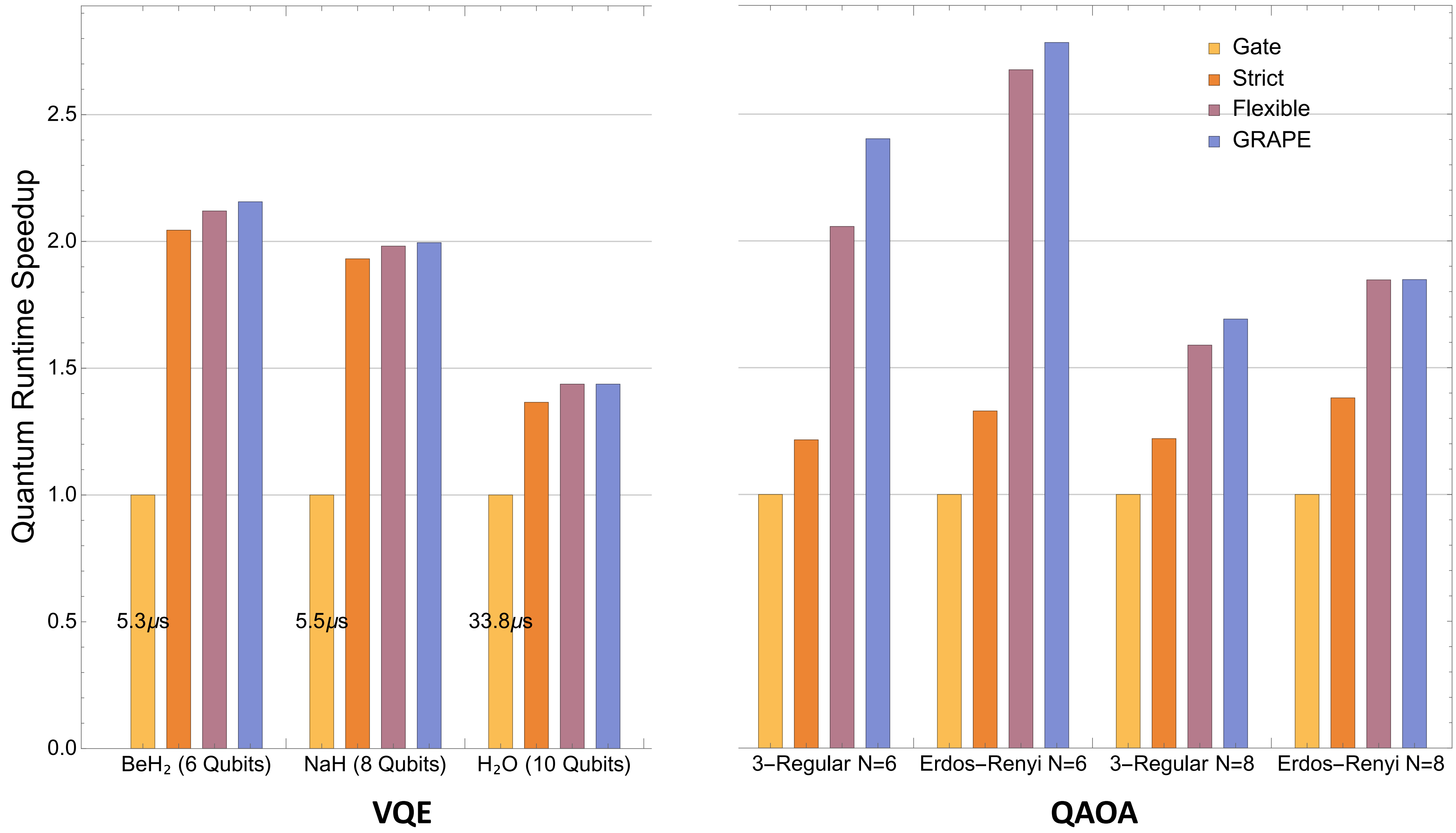


Quantum Runtime Speedup
0.0
0.5
1.0
1.5
2.0
Gate
Strict
Flexible
GRAPE
5.3μs
5.5μs
33.8μs
BeH2 (6 Qubits)
NaH (8 Qubits)
H2O (10 Qubits)

Results: QAOA

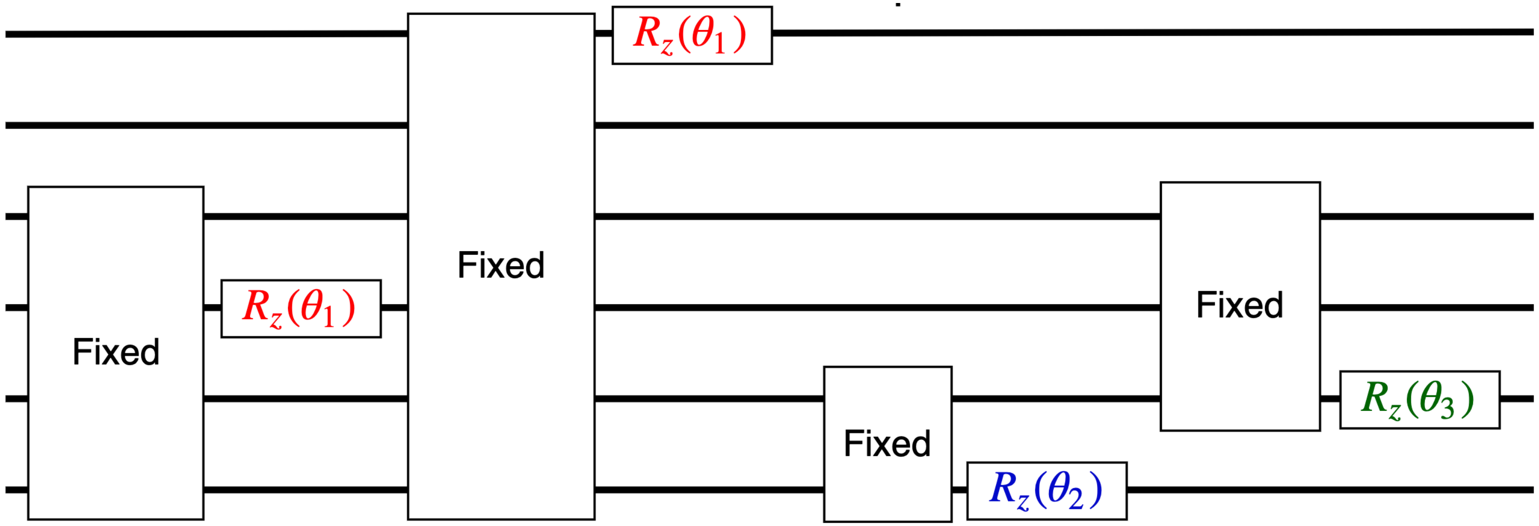


Results: Side-by-Side

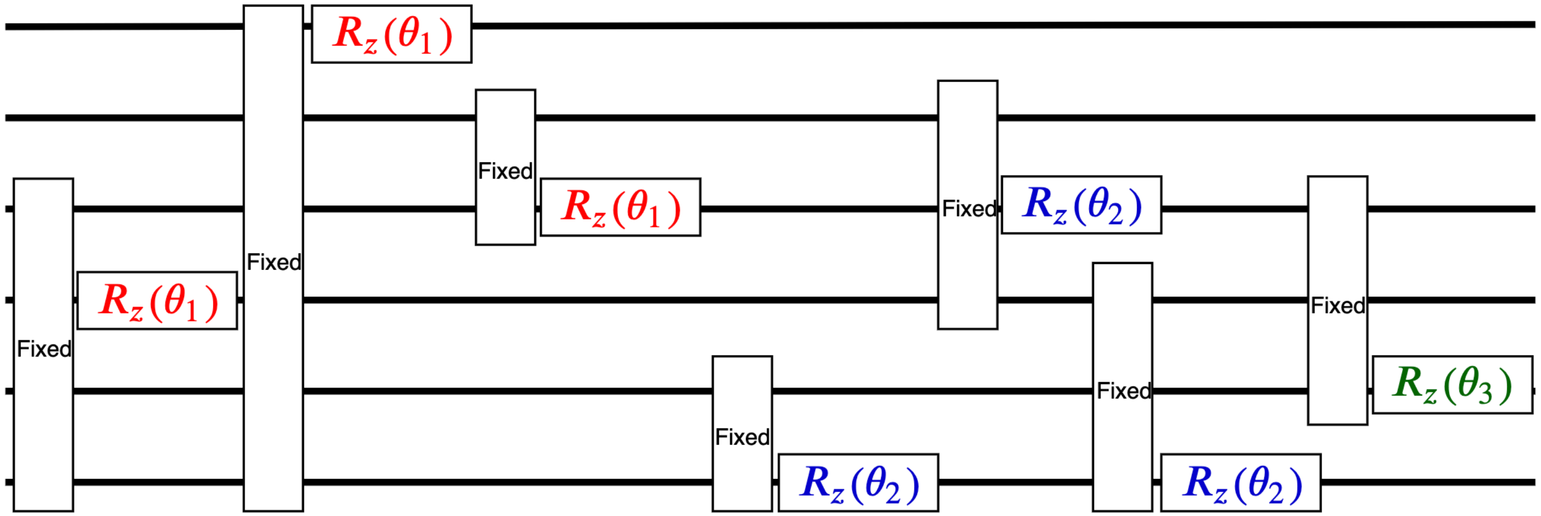


Results: Side-by-Side

VQE

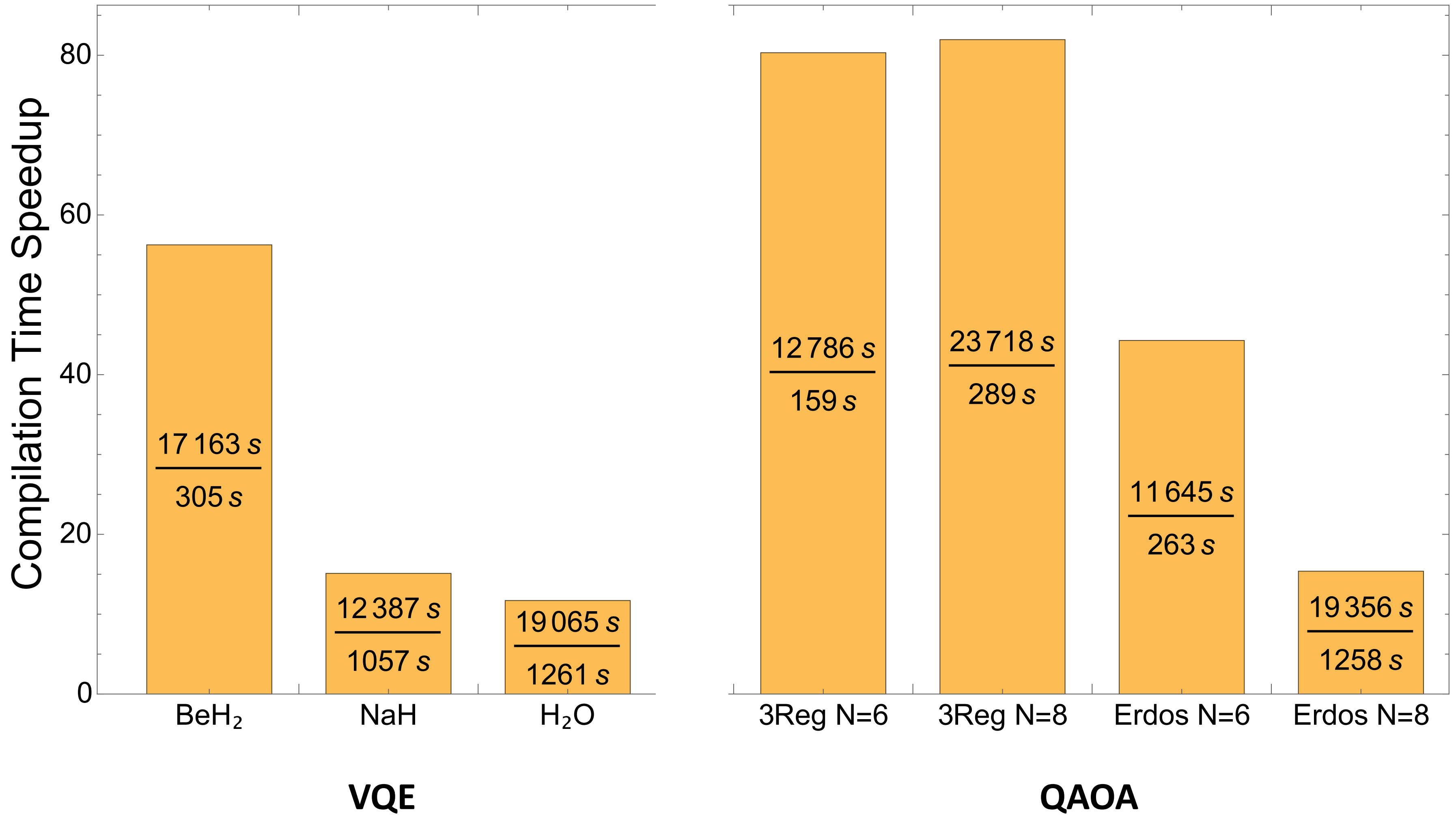


QAOA



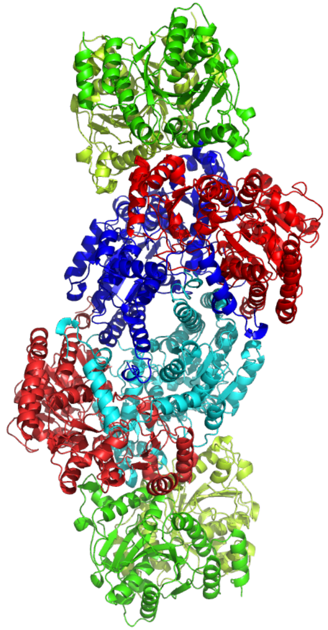
Results: Compilation Latency Reduction

Strict has 0 compilation time during runtime. For flexible:



Conclusions / Future Work

- 2x pulse speedups with minimal compilation latency
  - Strict has no latency
  - Flexible partial compilation has latency, but 10-80x lower than standard GRAPE.
- We propose experimental realizations, e.g. via OpenPulse
- Also a need for better (faster and smarter) GRAPE implementations



Thanks!

Partial Compilation of Variational Algorithms for Noisy Intermediate-Scale Quantum Machines

Pranav Gokhale[*]
University of Chicago

Yongshan Ding
University of Chicago

Thomas Propson
University of Chicago

Christopher Winkler
University of Chicago

Nelson Leung
University of Chicago

Yunong Shi
University of Chicago

David I. Schuster
University of Chicago

Henry Hoffmann
University of Chicago

Frederic T. Chong
University of Chicago

ABSTRACT

Quantum computing is on the cusp of reality with Noisy Intermediate-Scale Quantum (NISQ) machines currently under development and testing. Some of the most promising algorithms for these machines are *variational* algorithms that employ classical optimization coupled with quantum hardware to evaluate the quality of each candidate solution. Recent work used GRadient Descent Pulse Engineering (GRAPE) to translate quantum programs into highly optimized machine control pulses, resulting in a significant reduction in the execution time of programs. This is critical, as quantum machines can barely support the execution of short programs before failing.

However, GRAPE suffers from high compilation latency, which is untenable in variational algorithms since compilation is interleaved with computation. We propose two strategies for *partial compilation*, exploiting the structure of variational circuits to pre-compile optimal pulses for specific blocks of gates. Our results indicate significant pulse speedups ranging from 1.5x-3x in typical benchmarks, with only a small fraction of the compilation latency of GRAPE.

CCS CONCEPTS

• **Computer systems organization → Quantum computing**.

KEYWORDS

quantum computing, optimal control, variational algorithms

ACM Reference Format:
Pranav Gokhale, Yongshan Ding, Thomas Propson, Christopher Winkler, Nelson Leung, Yunong Shi, David I. Schuster, Henry Hoffmann, and Frederic T. Chong. 2019. Partial Compilation of Variational Algorithms for Noisy Intermediate-Scale Quantum Machines. In *The 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-52), October 12–16, 2019, Columbus, OH, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3352460.3358313

[*]Corresponding author: pranavgokhale@uchicago.edu

MICRO-52, October 12–16, 2019, Columbus, OH, USA

ACM ISBN 978-1-4503-6938-1/19/10…$15.00
https://doi.org/10.1145/3352460.3358313

1 INTRODUCTION

In the Noisy Intermediate-Scale Quantum (NISQ) era, we expect to operate hardware with hundreds or thousands of quantum bits (qubits), acted on by imperfect gates [42]. Moreover, connectivity in these NISQ machines will be sparse and qubits will have modest lifetimes. Given these limitations, NISQ era machines will not be able to execute large-scale quantum algorithms like Shor Factoring [45] and Grover Search [20], which rely on error correction that requires millions of qubits [38, 48].

However, recently, *variational algorithms* have been introduced that are well matched to NISQ machines. This new class of algorithms has a wide range of applications such as molecular ground state estimation [41], MAXCUT approximation [14], and prime factorization [2]. The two defining features of a variational algorithm are that:

(1) the algorithm complies with the constraints of NISQ hardware. Thus, the circuit for a variational algorithm should have modest requirements in qubit count (circuit width) and runtime (circuit depth / critical path).
(2) the quantum circuit for the algorithm is parametrized by a list of angles. These parameters are optimized by a classical optimizer over the course of many iterations. For this reason, variational algorithms are also termed as hybrid quantum-classical algorithms [42]. Typically, a classical optimizer that is robust to small amounts of noise (e.g. Nelder-Mead) is chosen [32, 41].

Standard non-variational quantum algorithms are fully specified at compile time and therefore can be fully optimized by static compilation tools as in previous work [23, 26]. By contrast, each iteration of a variational algorithm depends on the results of the previous iteration–hence, compilation must be interleaved through the computation. As even small instances of variational algorithms will require thousands of iterations [24], the compilation latency for each iteration therefore becomes a serious limitation. This feature of variational algorithms is a significant departure from previous non-variational quantum algorithms.

To cope with this limitation on compilation latency, past work on variational algorithms has performed compilation under the standard gate-based model. This methodology has the advantage of extremely fast compilation–a lookup table maps each gate to a sequence of machine-level control pulses so that compilation simply amounts to concatenating the pulses corresponding to each

arXiv:1909.07522

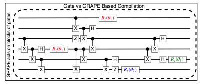
Partial compilation enables **2x pulse speedups** for quantum circuits, with reduced compilation latency.

Partial Compilation of Variational Algorithms for Noisy Intermediate-Scale Quantum Machines

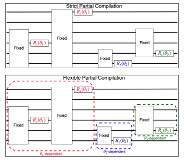
Pranav Gokhale, Yongshan Ding, Thomas Propson, Christopher Winkler, Nelson Leung, Yunong Shi, David I. Schuster, Henry Hoffmann, Frederic T. Chong

INTRO and METHODS

- Variational quantum circuits are costly in run-time (gates) or in compile time (pulses).

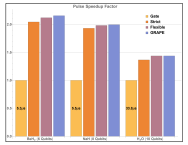


- Our partial compilation schemes minimize both.



RESULTS

- > 2x speedups for VQE and QAOA with minimal compilation latency

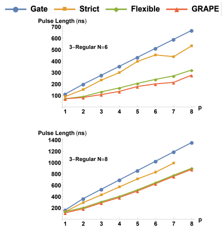


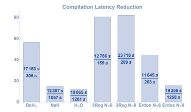
DISCUSSION

- Success probability decays exponential in pulse runtime—speedups are critical.
- Results persist for realistic pulses. Experimental efforts ongoing.

Partial Compil. Techniques

- Circuit blocking
- Parameter monotonicity
- Hyperparameter tuning





EPiQC
An NSF Expedition in Computing

Take a picture to download the full paper