INVITED PAPER

Resource-Efficient Quantum Computing by Breaking Abstractions

By 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 Martonosi, and Frederic T. Chong

ABSTRACT | Building a quantum computer that surpasses the computational power of its classical counterpart is a great engineering challenge. Quantum software optimizations can provide an accelerated pathway to the first generation of quantum computing (QC) 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 review, we point out that greater efficiency of QC systems can be achieved by breaking the abstractions between these layers. We review several works along this line, including two hardware-aware compilation optimizations that break the quantum instruction set architecture (ISA) abstraction and two error-correction/information-processing schemes that break the qubit abstraction. Last, we discuss several possible future directions.

KEYWORDS | Quantum computing (QC), software design, system analysis and design.

I. INTRODUCTION

Quantum computing (QC) has recently transitioned from a theoretical prediction to a nascent technology. With development of noisy intermediate-scale quantum (NISQ) devices, cloud-based quantum information processing (QIP) platforms with up to 53 qubits are currently accessible to the public. It has also been recently demonstrated by the Quantum Supremacy experiment on the Sycamore quantum processor, a 53-qubit QC device manufactured by Google, that quantum computers can outperform current classical supercomputers in certain computational tasks [7], although alternative classical simulations have been proposed that scale better [73], [74]. These developments suggest that the future of QC is promising. Nevertheless, there is still a gap between the ability and reliability of current QIP technologies and the requirements of the first useful QC applications. The gap is mostly due to the presence of qubit decoherence and systematic errors including gate errors, state preparation, and measurement (SPAM) errors. As an example, the best reported qubit decoherence time on a superconducting (SC) QIP platform is around 500 μs (meaning that in 500 μs, the probability of a logical 1 state staying unflipped drops to $1/e \approx 0.368$), the best error rate of 2-qubit gates is around 0.3%–1% in a device, measurement error of a single qubit is between 2% and 5% [1], [75]. In addition to the errors in the elementary operations, emergent error modes such

Manuscript received October 1, 2019; revised December 29, 2019 and March 23, 2020; accepted May 5, 2020. This work was supported in part by Enabling Practical-scale Quantum Computing (EPiQC), an NSF Expedition in Computing, under Grant CCF-1730449/1832377/1730082; in part by Software-Tailored Architectures for Quantum co-design (STAQ) under Grant NSF Phy-1818914; and in part by DOE under Grant DE-SC0020289 and Grant DE-SC0020331. Yunong Shi is funded in part by the NSF QISE-NET fellowship under grant number 1747426. Pranav Gokhale is supported by the Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program. This work was completed in part with resources provided by the University of Chicago Research Computing Center. *(Corresponding author: Frederic T. Chong.)*

Yunong Shi and **David I. Schuster** are with the Department of Physics, The University of Chicago, Chicago, IL 60637 USA.

Pranav Gokhale, **Jonathan M. Baker**, **Casey Duckering**, **Yongshan Ding**, and **Frederic T. Chong** are with the Department of Computer Science, The University of Chicago, Chicago, IL 60637 USA (e-mail: chong@cs.uchicago.edu).

Prakash Murali and **Margaret Martonosi** are with the Department of Computer Science, Princeton University, Princeton, NJ 08544 USA.

Natalie C. Brown and **Kenneth R. Brown** are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA.

Christopher Chamberland is with the AWS Center for Quantum Computing, Pasadena, CA 91125 USA, and also with the Institute for Quantum Information and Matter, California Institute of Technology, Pasadena, CA 91125 USA.

Ali Javadi-Abhari and **Andrew W. Cross** are with the IBM Thomas J. Watson Research Center, Ossining, NY 10598 USA.

Digital Object Identifier 10.1109/JPROC.2020.2994765

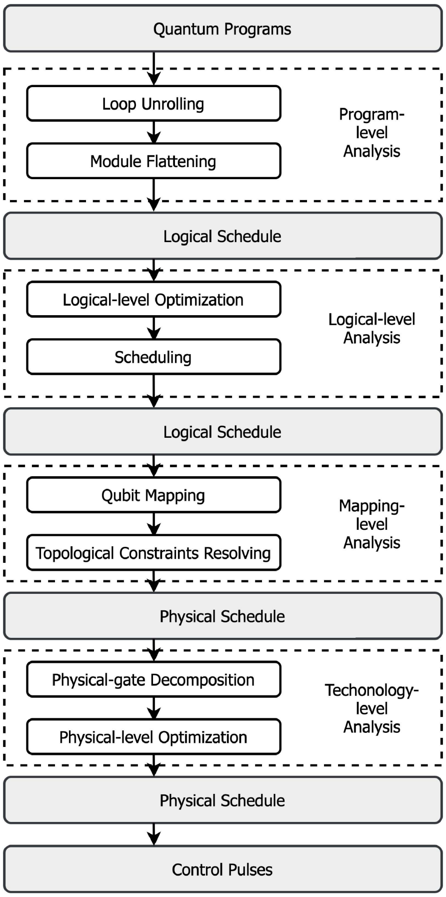


Fig. 1. ***Workflow of the QC stack roughly followed by current programming environments (e.g., Qiskit, Cirq, ScaffCC) based on the quantum circuit model.***

as crosstalk are reported to make significant contributions to the current noise level in quantum devices [18], [60]. With these sources of errors combined, we are only able to run quantum algorithms of very limited size on current QC devices.

Thus, it will require tremendous efforts and investment to solve these engineering challenges, and we cannot expect a definite timeline for its success. Because of the uncertainties and difficulties in relying on hardware breakthroughs, it will also be crucial in the near term to close the gap using higher-level quantum optimizations and software hardware codesign, which could maximally utilize noisy devices and potentially provide an accelerated pathway to real-world QC applications.

Currently, major quantum programming environments, including Qiskit [6] by IBM, Cirq [3] by Google, PyQuil [58] by Rigetti, and strawberry fields [66] by Xanadu, follow the quantum circuit model. These programming 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. 1. 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. Thus, this layered approach provides great portability and modularity. For example, the Qiskit compiler supports both the SC QIP platform and the trapped ion QIP platform as the backend (see Fig. 2). 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. SC qubits are macroscopic *LC* circuits placed inside dilution fridges of temperature near absolute zero. These qubits can be regarded as artificial atoms and are protected by a metal transmission line from environmental noise. For SC QIP platforms, qubit control is achieved by sending microwave pulses into the transmission line that surrounds the *LC* circuits to change the qubit state, and those operations are usually done within several hundreds of nanoseconds. On the other hand, trapped ion qubits are ions confined in the potential of electrodes in vacuum chambers. Trapped ion qubits have a much longer coherence time (>1 s) and a modulated laser beam is utilized (in addition to microwave pulse control) in performing quantum operations. The quantum gates are also much slower than that of SC qubits but the qubit connectivity (for 2-qubit gates) are much better. In Qiskit's early implementation, the hardware characteristics of the two QIP platforms are abstracted away in the quantum circuit model so that the higher level programming environment can work with both backends.

However, the abstractions introduced in the layered approach of current QC stacks restrict opportunities for cross-layer optimizations. For example, without accessing the lower level noise information, the compiler might not be able to properly optimize gate scheduling and qubit mapping with regard to the final fidelity. For near-term QC, maximal utilization of the scarce quantum resources and reconciling quantum algorithms with noisy devices is of more importance than to manage complexity of the classical control system. In this review, we propose a shift of the QC stack toward a more vertical integrated architecture. We point out that breaking the abstraction layers in the stack by exposing enough lower level details could substantially improve the quantum efficiency. This claim is not that surprising—there are many supporting examples from the classical computing world such as the emergence of application-specific architectures like the graphics processing unit (GPU) and the tensor processing unit (TPU). However, this view is often overlooked in the software/hardware design in QC.

We examine this methodology by looking at several previous works along this line. We first review two compilation optimizations that break the instruction set

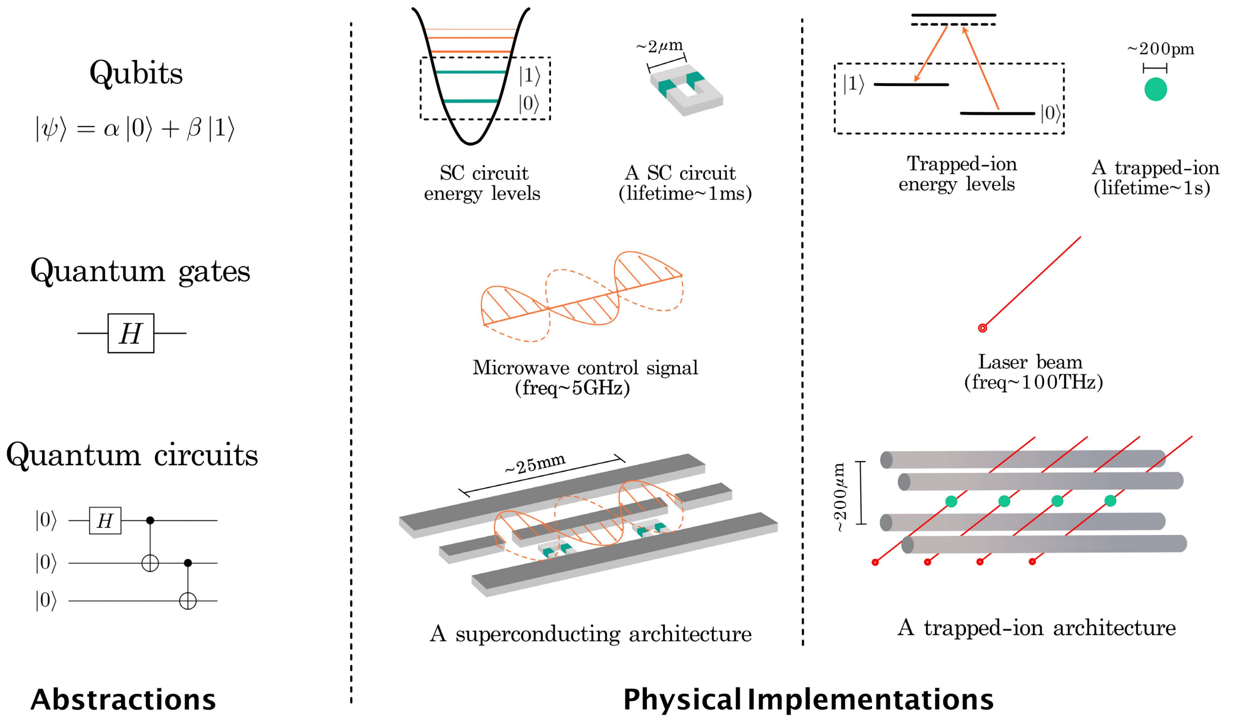

Qubits
$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$
Quantum gates
H
Quantum circuits
$|0\rangle$
$|0\rangle$
$|0\rangle$
H
Abstractions
$|1\rangle$
$|0\rangle$
SC circuit energy levels
~2μm
A SC circuit (lifetime~1ms)
Microwave control signal (freq~5GHz)
~25mm
A superconducting architecture
$|1\rangle$
$|0\rangle$
Trapped-ion energy levels
~200pm
A trapped-ion (lifetime~1s)
Laser beam (freq~100THz)
~200μm
A trapped-ion architecture
Physical Implementations

Fig. 2. *Same abstractions in the QC stack on the logical level can be mapped to different physical implementations. Here, we take the SC QIP platform and the trapped ion QIP platform as examples of the physical implementations. (Left) In the quantum circuit model, both SC qubits and trapped-ion qubits are abstracted as two-level quantum systems and their physical operations are abstracted as quantum gates, even though these two systems have different physical properties. (Middle) SC qubits are SC circuits placed inside a long, metal transmission line. The apparatus requires a dilution fridge of temperature near absolute zero. The orange standing waves are oscillations in the transmission line, which are driven by external microwave pulses and used to control the qubit states. (Right) Trapped ion qubits are confined in the potential of cylindrical electrodes. Modulated laser beam can provide elementary quantum operations for trapped ion qubits. The apparatus is usually contained inside a vacuum chamber of pressure around 10^{-8} Pa. The two systems require different high-level optimizations for better efficiency due to their distinct physical features.*

architecture (ISA) abstraction by exposing pulse level information (see Section II) and noise information (see Section III). Then, we discuss an information processing scheme that improves general circuit latency by exposing the third energy level of the underlying physical space, that is, breaking the qubit abstraction using qutrits (see Section IV). Then, we discuss the Gottesman–Kitaev–Preskill (GKP) qubit encoding in a quantum harmonic oscillator (QHO) that exposes error information in the form of small shifts in the phase space to assist the upper level error mitigation/correction procedure (see Section V).

At last, we envision several future directions that could further explore the idea of breaking abstractions and assist the realization of the first quantum computers for real-world applications.

II. BREAKING THE ISA ABSTRACTION USING PULSE-LEVEL COMPILATION

In this section, we describe a quantum compilation methodology proposed in [28] and [67] that achieves an average of 5× speedup in terms of generated circuit latency, by employing the idea of breaking the ISA abstraction and compiling directly to control pulses.

A. Quantum Compilation

Since the early days of QC, quantum compilation 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 for a given unitary matrix. The celebrated Solovay–Kitaev theorem [34] states that such synthesis is always possible if a universal set of quantum gates is given. Now the term of quantum compilation is used more broadly and almost all stages in Fig. 1 can be viewed as part of the quantum compilation process.

There are many indications that current quantum compilation stack (see Fig. 1) is highly inefficient. First, current circuit synthesis algorithms are far from saturating (or being closed to) the asymptotic lower bound in the general case [34], [49]. Also, the formulated circuit synthesis problem is based on the fundamental abstraction of quantum ISA (see Section II-B) 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). 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 QC stack, thus underlying quantum devices' computing ability and wastes precious quantum coherence. While improving the computing efficiency is always valuable, improving QC 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 [28] and [66], 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 corresponds 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 section to introduce this compilation methodology, starting with defining some basic concepts.

B. Quantum ISA

In the QC stack, a restricted set of 1- and 2-qubit quantum instructions are provided for describing the high-level quantum algorithms, analogous to the ISA abstraction in classical computing. In this article, we call this instruction set the logical ISA. The 1-qubit gates in the logical ISA include the Pauli gates, $P = \{X, Y, Z\}$. It also includes the Hadamard H gate, the symbol in the circuit model of which is given as an example in Fig. 2 on the left column. The typical 2-qubit instruction in the logical instruction set is the controlled-NOT (CNOT) gate, which flips the state of the target qubit based on the state of the control qubit.

However, usually QC devices does not directly support the logical ISA. Based on the system characteristics, we can define the physical ISA that can be directly mapped to the underlying control signals. For example, SC devices typically has cross-resonance (CR) gate or iSWAP gate as their intrinsic 2-qubit instruction, whereas for trapped-ion devices the intrinsic 2-qubit instruction can be the Mølmer–Sørensen gate or the controlled phase gate.

C. Quantum Control

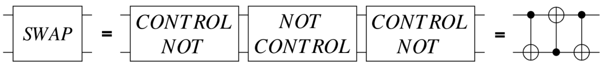
As shown in Fig. 2 and discussed in Section I, underlying physical operations in the hardware such as microwave control pulses and modulated laser beam are abstracted as quantum instructions. A quantum instruction is simply as prefined control pulse sequences.

The underlying evolution of the quantum system is continuous and so are the control signals. The continuous control signals offer much richer and flexible controllability than the quantum ISA. The control pulses can drive the QC hardware to a desired quantum states by varying a system-dependent and time-dependent quantity called the Hamiltonian. The Hamiltonian of a system determines the evolution path of the quantum states. The ability to engineer real-time system Hamiltonian allows us to navigate the quantum system to the quantum state of interest through generating accurate control signals. Thus, quantum computation can be done by constructing a quantum system in which the system Hamiltonian evolves in a way that aligns with a QC task, producing the computational result with high probability upon final measurement of the qubits. In general, the path to a final quantum state is not unique, and finding the optimal evolution path is a very important but challenging problem [25], [39], [62].

D. Mismatch Between ISA and Control

Being hardware-agnostic, the quantum operation sequences composed by logical ISA have limited freedom in terms of controllability and usually will not be mapped to the optimal evolution path of the underlying quantum system, thus there is a mismatch between the ISA and low-level quantum control. With two simple examples, we demonstrate this mismatch.

1) We can consider the instruction sequence consists of a CNOT gate followed by an X gate on the control bit. In current compilation workflow, these two logical gates will be further decomposed into the physical ISA and be executed sequentially. However, on SC QIP platforms, the microwave pulses that implement these two instructions could in fact be applied simultaneously (because of their commutativity). Even the commutativity can be captured by the ISA abstraction, in the current compilation workflow, the compiled control signals are suboptimal.
2) SWAP gate is an important quantum instruction for circuit mapping. The SWAP operation is usually decomposed as three CNOT operations, as realized in the circuit below. This decomposition could be thought of the implementation of in-place memory SWAPs with three alternating XORs for classical computation. However, for systems like quantum dots [41], the SWAP operation is directly supported by applying particular constant control signals for a certain period of time. In this case, this decomposition of SWAP into three CNOTs introduces substantial overhead.



In experimental physics settings, equivalences from simple gate sequences to control pulses can be hand optimized [61]. However, when circuits become larger and more complicated, this kind of hand optimization becomes less efficient and the standard decomposition becomes less favorable, motivating a shift toward numerical optimization methods that are not limited by the ISA abstraction.

E. Quantum Optimal Control

Quantum optimal control (QOC) theory provides an alternative in terms of finding the optimal evolution path for the quantum compilation tasks. QOC algorithms typically perform analytical or numerical methods for this optimization, among which, gradient ascent methods, such as the GRadient Ascent Pulse Engineering (GRAPE) [15], [33] algorithm, are widely used. The basic idea of GRAPE is as follows: for optimizing the control signals of M parameters $(u_1, \dots, u_M)$ for a target quantum state, in every iteration, GRAPE minimizes the deviation of the system evolution by calculating the gradient of the final fidelity with respect to the M control parameters in the M-dimensional space. Then GRAPE will update the parameters in the direction of the gradient with adaptive step size [15], [33], [39]. With a large number of iterations, the optimized control signals are expected to converge and find optimized pulses.

In [65], we utilize GRAPE to optimize our aggregated instructions that are customized for each quantum circuit as opposed to selecting instructions from a predefined pulse sequences. However, one disadvantage of numerical methods like GRAPE is that the running time and memory use grow exponentially with the size of the quantum system for optimization. In our work, we are able to use GRAPE for optimizing quantum systems of up to 10 qubits with the GPU-accelerated optimal control unit [39]. As shown in our result, the limit of 10 qubits does not put restrictions on the result of our compilation methodology.

F. Pulse-Level Optimization: A Motivating Example

Next, we will illustrate the workflow of our compilation methodology with a circuit instance of the quantum approximate optimization algorithm (QAOA) for solving the MAXCUT problem on the triangle graph (see Fig. 3).[1] This QAOA circuit with logical ISA (or variants of it up to single qubit gates) can be reproduced by most existing quantum compilers. This instance of the QAOA circuit is generated by the ScaffCC compiler, as shown in Fig. 3(a). We assume this circuit is executed on an SC architecture with 1-D nearest neighbor qubit connectivity. A SWAP instruction is inserted in the circuit to satisfy the linear qubit connectivity constraints.

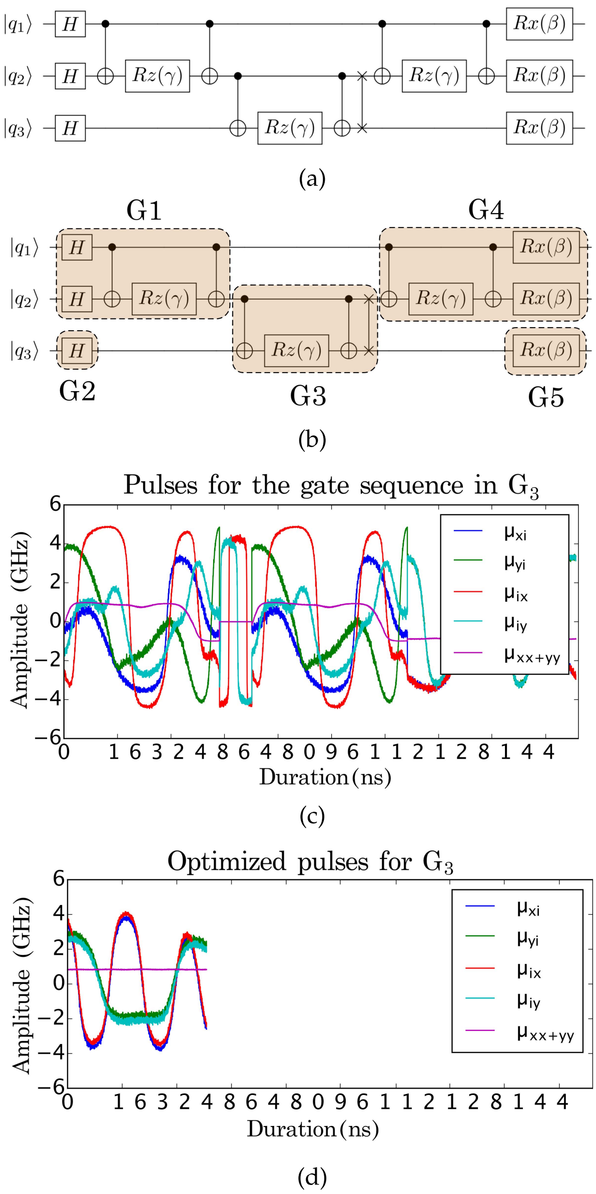


Fig. 3. *Example of a QAOA circuit. (a) QAOA circuit with the logical ISA. (b) QAOA circuit with aggregated instructions. (c) Generated control pulses for G_3 in the ISA-based compilation. (d) Control pulses for G_3 from aggregated instructions based compilation. Each curve is the amplitude of a relevant control signal. The pulse sequences in (d) provides a 3× speedup comparing to the pulse sequences in (c). Pulse sequences reprinted with permission from [65].*

On the other hand, our compiler generates the aggregated instruction set G_1–G_5 as illustrated in Fig. 3(b) automatically, and uses GRAPE to produce highly optimized pulse sequences for each aggregated instruction. In this minimal circuit instance, our compilation method reduces the total execution time of the circuit by about $2.97\times$ compared to compilation with restricted ISA. Fig. 3(c) and (d) shows the generated pulses for G_3 with ISA-based compilation and with our aggregated instruction based, pulse-level optimized compilation.

G. Optimized Pulse-Level Compilation Using Gate Aggregation: The Workflow

Now, we give a systematic view of the workflow of our compiler. First, at the program level, our compiler

[1] The angle parameters γ and β can be determined by variational methods [44] and are set to 5.67 and 1.26.

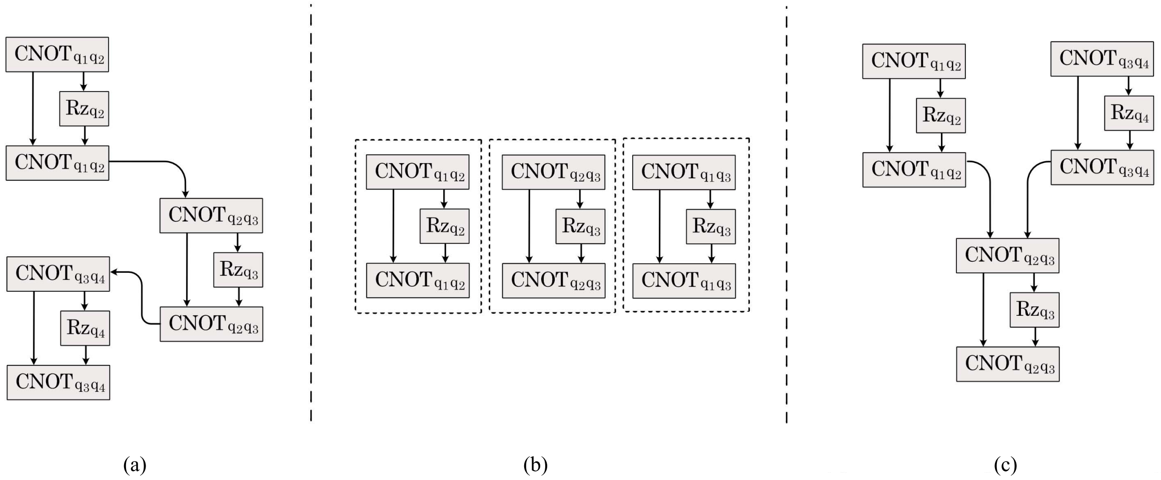


Fig. 4. ***Example in Fig. 5 in the form of GDG. (a) Input GDG. (b) Commutativity detection. (c) Commutativity-aware scheduling.***

performs module flattening and loop unrolling to produce the quantum assembly (QASM), which represents a schedule of the logical operations. Next, the compiler enters the commutativity detection phase. Different from the ISA-based approach, in this phase, our compilation process converts the QASM code to a more flexible logical schedule that explores the commutativity between instructions. To further explore the commutativity in the schedule, the compiler aggregates instructions in the schedule to produce a new logical schedule with instructions that represents diagonal matrices (and are of high commutativity). Then the compiler enters the scheduling and mapping phase. Because of commutativity awareness, our compiler can generate a much more efficient logical schedule by rearranging the aggregated instructions with high commutativity. The logical schedule is then converted to a physical schedule after the qubit mapping stage. Then the compiler generates the final aggregated instructions for pulse optimization and use GRAPE for producing the corresponding control pulses. The goal of the final aggregation is to find the optimal instruction set that produces the lowest-latency control pulses while preserving the parallelism in the circuit aggregations that are small as much as possible. Finally, our compiler outputs an optimized physical schedule along with the corresponding optimized control pulses. Fig. 4 shows the gate dependence graph (GDG) of the QAOA circuit in Fig. 5 in different compilation stages. Next, we walk through the compilation backend with this example, starting from the commutativity detection phase.

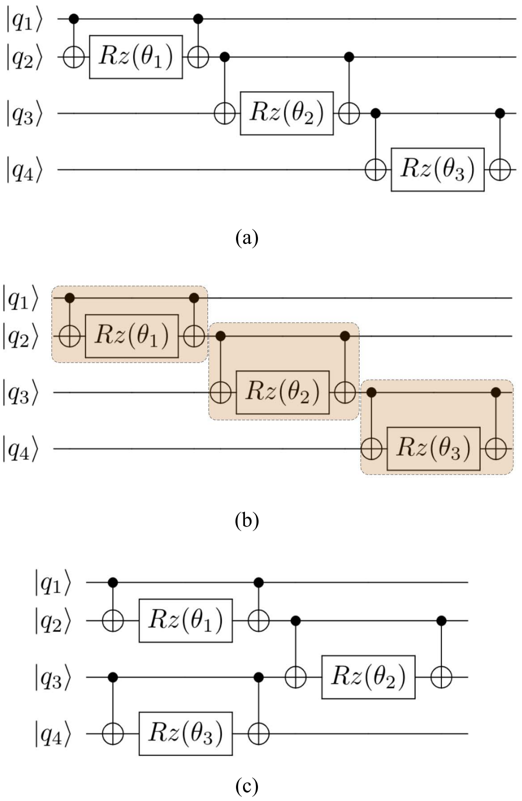


Fig. 5. ***Example of CLS. With commutativity detected, the circuit depth can be shortened. (a) Input circuit. (b) Commutativity detection. (c) Commutativity-aware scheduling.***

1) *Commutativity Detection:* In the commutativity detection phase, the false dependence between commutative instructions are removed and the GDG is restructureed. This is because if a pair of gates commutes, the gates can be scheduled in either order. Also, it can be further noticed that, in many NISQ quantum algorithms, it is ubiquitous that for instructions within an instruction block to not commute, but for the full instruction block to commute with each other [19], [37]. As an example, in Fig. 5, the CNOT-Rz-CNOT instruction blocks commute with each other because these blocks correspond to diagonal unitary matrices. However, each individual instruction in these

circuit blocks does not commute. Thus, after aggregating these instructions, the compiler is able to schedule new aggregated instructions in any order, which is impossible before. This commutativity detection procedure opens up opportunities for more efficient scheduling.

2) Scheduling and Mapping:

a) Commutativity-aware logical scheduling (CLS): In our scheduling phase, our logical scheduling algorithm is able to fully utilize the detected commutativity in the last compilation phase. The CLS iteratively schedules the available instructions on each qubits. At each iteration, the CLS draws instruction candidates that can be executed in the earliest time step to schedule.

b) Qubit mapping: In this phase of the compilation, the compiler transform the circuit to a form that respect the topological constraints of hardware connectivity [43]. To conform to the device topology, the logical 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 [32]. Once the initial mapping is generated, 2-qubit operations between nonneighboring qubits are prepended with a sequence of SWAP rearrangements that move the control and target qubits to be adjacent.

3) Instruction Aggregation: In this phase, the compiler iterates with the optimal control unit to generate the final aggregated instructions for the circuit. Then, the optimal control unit optimizes each instruction individually with GRAPE.

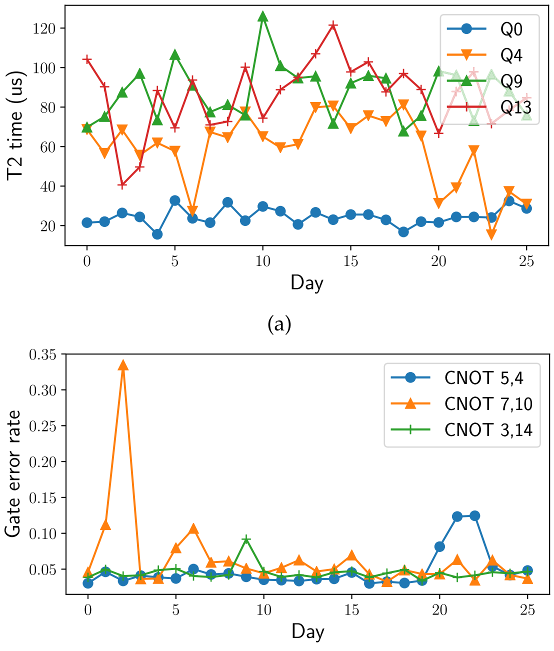
4) Physical Execution: Finally, the circuit will be scheduled again using the CLS from Section II-G2, the physical schedules will be sent to the control unit of the underlying quantum hardware and trigger the optimized control pulses at appropriate timing and the physical execution.

H. Discussion

In [65], we selected nine important quantum/classical-quantum hybrid algorithms in the NISQ era as our benchmarks. Across all nine benchmarks, our compilation scheme achieves a geometric mean of 5.07× pulse time reduction comparing to the standard gate-based compilation. The result in [65] indicates that addressing the mismatch between quantum gates and the control pulses by breaking the ISA abstraction can greatly improve the compilation efficiency. Going beyond the ISA-based compilation, this article opens up a door to new QC system designs.

III. BREAKING THE ISA ABSTRACTION USING NOISE-ADAPTIVE COMPILATION

In recent years, QC compute stacks have been developed using abstractions inspired from classical computing. The ISA is a fundamental abstraction which defines the interface between the hardware and software. The ISA abstraction allows software to execute correctly on any hardware which implements the interface. This enables application portability and decouples hardware and software development.



(a)

(b)

Fig. 6. ***Daily variations in qubit coherence time (larger is better) and gate error rates (lower is better) for selected qubits and gates in IBM's 16-qubit system. The most or least reliable system elements change across days. (a) Coherence time (T2). (b) CNOT gate error rate.***

For QC systems, the hardware–software interface is typically defined as a set of legal instructions and the connectivity topology of the qubits [14], [20]–[22], [58]—it does not include information about qubit quality, gate fidelity, or micro-operations used to implement the ISA instructions. While technology independent abstractions are desirable in the long run, our work [46], [47] reveals that such abstractions are detrimental to program correctness in NISQ quantum computers. By exposing microarchitectural details to software and using intelligent compilation techniques, we show that program reliability can be improved significantly.

A. Noise Characteristics of QC Systems

QC systems have spatial and temporal variations in noise due to manufacturing imperfections, imprecise qubit control, and external interference. To motivate the necessity for breaking the ISA abstraction barrier, we present real-system statistics of hardware noise in systems from three leading QC vendors—IBM, Rigetti, and University of Maryland. IBM and Rigetti systems use SC qubits [10], [12] and the University of Maryland (UMD) uses

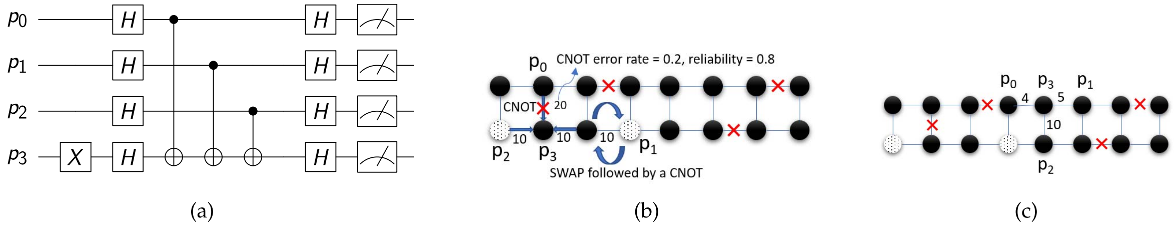


Fig. 7. *(a) IR of the Bernstein-Vazirani algorithm (BV4). Each horizontal line represents a program qubit. X and H are single qubit gates. The CNOT gates from each qubit p_{0--2} to p_3 are marked by vertical lines with XOR connectors. The readout operation is indicated by the meter. (b) Qubit layout in IBMQ16, a naive mapping of BV4 onto this system. The black circles denote qubits and the edges indicate hardware CNOT gates. The edges are labeled with CNOT gate error ($\times 10^{-2}$). The hatched qubits and crossed gates are unreliable. In this mapping, a SWAP operation is required to perform the CNOT between p_1 and p_3 and error-prone operations are used. (c) Mapping for BV4 where qubit movement is not required and unreliable qubits and gates are avoided.*

trapped ion qubits [16]. The gates in these systems are periodically calibrated and their error rates are measured.

Fig. 6 shows the coherence times and 2-qubit gate error rates in IBM's 16-qubit system (ibmnamefull). From daily calibration logs we find that, the average qubit coherence time is 40 μs, 2-qubit gate error rate is 7%, readout error rate is 4%, and single qubit error rate is 0.2%. The 2-qubit and readout errors are the dominant noise sources and vary up to 9× across gates and calibration cycles. Rigetti's systems also exhibit error rates and variations of comparable magnitude. These noise variations in SC systems emerge from material defects due to lithographic manufacturing, and are expected in the future systems also [35], [36].

Trapped ion systems also have noise fluctuations even though the individual qubits are identical and defect-free. On a 5-qubit trapped ion system from UMD, we observed up to $3\times$ variation in the 2-qubit gate error rates because of fundamental challenges in qubit control using lasers and their sensitivity to motional mode drifts from temperature fluctuations.

We found that these microarchitectural noise variations dramatically influence program correctness. When a program is executed on a noisy QC system, the results may be corrupted by gate errors, decoherence, or readout errors on the hardware qubits used for execution. Therefore, it is crucial to select the most reliable hardware qubits to improve the success rate of the program (the likelihood of correct execution). The success rate is determined by executing a program multiple times and measuring the fraction of runs that produce the correct output. High success rate is important to ensure that the program execution is not dominated by noise.

B. Noise-Adaptive Compilation: Key Ideas

Our work breaks the ISA abstraction barrier by developing compiler optimizations which use hardware calibration data. These optimizations boost the success rate a program run by avoiding portions of the machine with poor coherence time and operational error rates.

We first review the key components in a QC compiler. The input to the compiler is a high-level language program (Scaffold in our framework) and the output is machine executable assembly code. First, the compiler converts the program to an intermediate representation (IR) composed of single and 2-qubit gates by decomposing high-level QC operations, unrolling all loops and inlining all functions. Fig. 7(a) shows an example IR. The qubits in the IR (program qubits) are mapped to distinct qubits in the hardware, typically in a way that reduces qubit communication. Next, gates are scheduled while respecting data dependences. Finally, on hardware with limited connectivity, such as SC systems, the compiler inserts SWAP operations to enable 2-qubit operations between nonadjacent qubits.

Fig. 7(a) and (b) shows two compiler mappings for a 4-qubit program on IBM's 16-qubit system. In the first mapping, the compiler must insert SWAPs to perform the 2-qubit gate between p_1 and p_3. Since SWAP operations are composed of three 2-qubit gates, they are highly error prone. In contrast, the second mapping requires no SWAPs because the qubits required for the CNOTs are adjacent. Although SWAP optimizations can be performed using the device ISA, the second mapping is also noise-optimized, that is, it uses qubits with high coherence time and low operational error rates. By considering microarchitectural noise characteristics, our compiler can determine such noise-optimized mappings to improve the program success rate.

We developed three strategies for noise optimization. First, our compiler maps program qubits onto hardware locations with high reliability, based on the noise data. We choose the initial mapping based on 2-qubit and readout error rates because they are the dominant sources of error. Second, to mitigate decoherence errors, all gates are scheduled to finish before the coherence time of the hardware qubits. Third, our compiler optimizes the reliability of SWAP operations by minimizing the number of SWAPs whenever possible and performing SWAPs along reliable hardware paths.

C. Implementation Using Satisfiability Modulo Theory (SMT) Optimization

Our compiler implements the above strategies by finding the solution to a constrained optimization problem

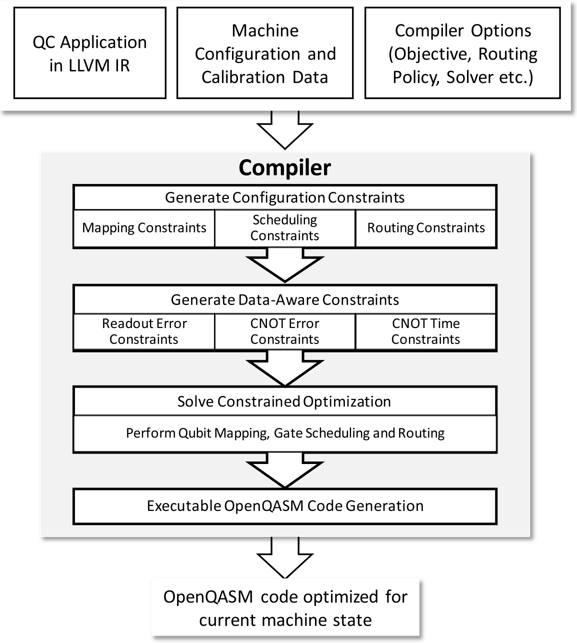

QC Application in LLVM IR
Machine Configuration and Calibration Data
Compiler Options (Objective, Routing Policy, Solver etc.)
Compiler
Generate Configuration Constraints
Mapping Constraints
Scheduling Constraints
Routing Constraints
Generate Data-Aware Constraints
Readout Error Constraints
CNOT Error Constraints
CNOT Time Constraints
Solve Constrained Optimization
Perform Qubit Mapping, Gate Scheduling and Routing
Executable OpenQASM Code Generation
OpenQASM code optimized for current machine state

Fig. 8. ***Noise-adaptive compilation using SMT optimization. Inputs are a QC program IR, details about the hardware qubit configuration, and a set of options, such as routing policy and solver options. From these, compiler generates a set of appropriate constraints and uses them to map program qubits to hardware qubits and schedule operations. The output of the optimization is used to generate an executable version of the program.***

using an SMT solver. The variables and constraints in the optimization encode program information, device topology constraints, and noise information. The variables express the choices for program qubit mappings, gate start times, and routing paths. The constraints specify that qubit mappings should be distinct, the schedule should respect program dependences, and that routing paths should be nonoverlapping. Fig. 8 summarizes the optimization-based compilation pipeline for IBMQ16.

The objective of our optimization is to maximize the success rate of a program execution. Since the success rate can be determined only from a real-system run, we model it at compile time as the program reliablity. We define the reliability of a program as the product of reliability of all gates in the program. Although this is not a perfect model for the success rate, it serves as a useful measure of overall correctness [7], [40]. For a given mapping, the solver determines the reliability of each 2-qubit and readout operation and computes an overall reliability score. The solver maximizes the reliability score over all mappings by tracking and adapting to the error rates, coherence limits, and qubit movement based on program qubit locations.

In practice, we use the Z3 SMT solver to express and solve this optimization. Since the reliability objective is a nonlinear product, we linearize the objective by optimizing for the additive logarithms of the reliability scores of each gate. We term this algorithm as R-SMT*. The output of the SMT solver is used to create machine executable code in the vendor-specified assembly language.

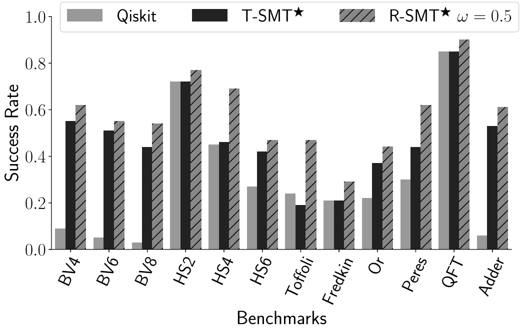


Fig. 9. ***Measured success rate of R-SMT* compared to Qiskit and T-SMT*. (Of 8192 trials per execution, success rate is the percentage that achieve the correct answer in real-system execution.) ω is a weight factor for readout error terms in the R-SMT* objective, 0.5 is equal weight for CNOT and readout errors. R-SMT* obtains higher success rate than Qiskit because it adapts the qubit mappings according to dynamic error rates and also avoids unnecessary qubit communication.***

D. Real-System Evaluation

We present real-system evaluation on IBMQ16. Our evaluation uses 12 common QC benchmarks, compiled using R-SMT* and T-SMT* which are variants of our compiler and IBM's Qiskit compiler (version 0.5.7) [6] which is the default for this system. R-SMT* optimizes the reliability of the program using hardware noise data. T-SMT* optimizes the execution time of the program considering real-system gate durations and coherence times, but not operational error rates. IBM Qiskit is also noise-unaware and uses randomized algorithms for SWAP optimization. For each benchmark and compiler, we measured the success rate on IBMQ16 system using 8192 trials per program. A success rate of 1 indicates a perfect noise-free execution.

Fig. 9 shows the success rate for the three compilers on all the benchmarks. R-SMT* has higher success rate than both baselines on all benchmarks, demonstrating the effectiveness of noise-adaptive compilation. Across benchmarks R-SMT* obtains geomean $2.9\times$ improvement over Qiskit, with up to $18\times$ gain. Fig. 10 shows the mapping used by Qiskit, T-SMT*, and R-SMT* for BV4. Qiskit places qubits in a lexicographic order without considering CNOT and readout errors and incurs extra swap operations. Similarly, T-SMT* is also unaware of noise variations across the device, resulting in mappings which use unreliable hardware. R-SMT* outperforms these baselines because it maximizes the likelihood of reliable execution by leveraging microarchitectural noise characteristics during compilation.

Full results of our evaluation on seven QC systems from IBM, Rigetti, and UMD can be found in [47] and [48].

E. Discussion

Our work represents one of the first efforts to exploit hardware noise characteristics during compilation. We developed optimal and heuristic techniques for noise

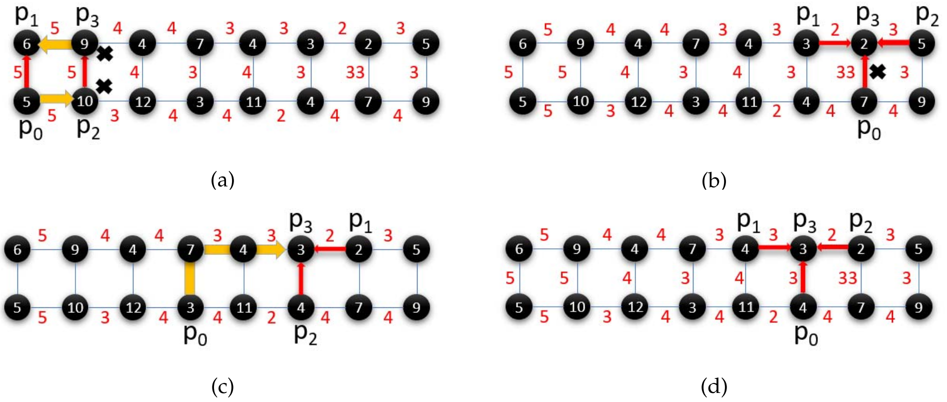


Fig. 10. *For real data/experiment, on IBMQ16, qubit mappings for Qiskit and our compiler with three optimization objectives, varying the type of noise-awareness. The edge labels indicate the CNOT gate error rate ($\times 10^{-2}$), and the node labels indicate the qubit's readout error rate ($\times 10^{-2}$). The thin red arrows indicate CNOT gates. The thick yellow arrows indicate SWAP operations. ω is a weight factor for readout error terms in the R-SMT* objective. (a) Qiskit finds a mapping which requires SWAP operations and uses hardware qubits with high readout errors. (b), T-SMT* finds a a mapping which requires no SWAP operations, but it uses an unreliable hardware CNOT between p_3 and p_0. (c) Program qubits are placed on the best readout qubits, but p_0 and p_3 communicate using swaps. (d) R-SMT* finds a mapping which has the best reliability where the best CNOTs and readout qubits are used. It also requires no SWAP operations. (a) IBM Qiskit. (b) T-SMT*:Optimize duration without error data. (c) R-SMT* ($\omega = 1$): Optimize readout reliability. (d) R-SMT* ($\omega = 0.5$): Optimize CNOT$+$readout reliability.*

adaptivity and performed comprehensive evaluations on several real QC systems [47]. We also developed techniques to mitigate crosstalk, another major source of errors in QC systems, using compiler techniques that schedule programs using crosstalk characterization data from the hardware [48]. In addition, our techniques are already being used in industry toolflows [54], [59]. Recognizing the importance of efficient compilation, other research groups have also recently developed mapping and routing heuristics [11], [72] and techniques to handle noise [67], [68].

Our noise-adaptivity optimizations offer large gains in success rate. These gains mean the difference between executions which yield correct and usable results and executions where the results are dominated by noise. These improvements are also multiplicative against benefits obtained elsewhere in the stack and will be instrumental in closing the gap between near-term QC algorithms and hardware. Our work also indicates that it is important to accurately characterize hardware and expose characterization data to software instead of hiding it behind a device-independent ISA layer. Additionally, our work also proposes that QC programs should be compiled once-per-execution using the latest hardware characterization data to obtain the best executions.

Going beyond noise characteristics, we also studied the importance of exposing other microarchitectural information to software. We found that when the compiler has access to the native gates available in the hardware (micro operations used to implement ISA-level gates), it can further optimize programs and improve success rates. Overall, our work indicates that QC machines are not yet ready for technology independent abstractions that shield the software from hardware. Bridging the information gap between software and hardware by breaking abstraction barriers will be increasingly important on the path toward practically useful NISQ devices.

IV. BREAKING THE QUBIT ABSTRACTION VIA THE THIRD ENERGY LEVEL

Although quantum computation is typically expressed with the two-level binary abstraction of qubits, the underlying physics of quantum systems are not intrinsically binary. Whereas classical computers operate in binary states at the physical level (e.g., clipping above and below a threshold voltage), quantum computers have natural access to an infinite spectrum of discrete energy levels. In fact, hardware must actively suppress higher level states in order to realize an engineered two-level qubit approximation. In this sense, using three-level qutrits (quantum trits) is simply a choice of including an additional discrete energy level within the computational space. Thus, it is appealing to explore what gains can be realized by breaking the binary qubit abstraction.

In prior work on qutrits (or more generally, d-level qudits), researchers identified only constant factor gains from extending beyond qubits. In general, this prior work [53] has emphasized the information compression advantages of qutrits. For example, N qubits can be expressed as $(N/\log_2(3))$ qutrits, which leads to $\log_2(3) \approx 1.6$-constant factor improvements in runtimes.

Recently, however, our research group demonstrated a novel qutrit approach that leads to exponentially faster runtimes (i.e., shorter in circuit depth) than qubit-only approaches [26], [27]. The key idea underlying the approach is to use the third state of a qutrit as temporary storage. Although qutrits incur higher per-operation error rates than qubits, this is compensated by dramatic reductions in runtimes and quantum gate counts. Moreover, our approach applies qutrit operations only in an intermediary stage: the input and output are still qubits, which is important for initialization and measurement on practical quantum devices [56], [57].

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

A. Qutrit-Assisted AND Gate

We develop the intuition for how qutrits can be useful by considering the example of constructing an AND gate. In the framework of QC, which requires reversibility, AND is not permitted directly. For example, consider the output of 0 from an AND gate with two inputs. With only this information about the output, the value of the inputs cannot be uniquely determined (00, 01, and 10 all yield an AND output of 0). However, these operations can be made reversible by the addition of an extra, temporary workspace bit initialized to 0. Using a single additional such as ancilla, the AND operation can be computed reversibly as in Fig. 11. Although this approach works, it is expensive—in order to decompose the Toffoli gate in Fig. 11 into hardware-implementable one- and two-input gates, it is decomposed into at least six CNOT gates.

However, if we break the qubit abstraction and allow occupation of a higher qutrit energy level, the cost of the Toffoli AND operation is greatly diminished. Before proceeding, we review the basics of qutrits, which have three computational basis states: $|0\rangle$, $|1\rangle$, and $|2\rangle$. A qutrit state $|\psi\rangle$ may be represented analogously to a qubit as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle + \gamma|2\rangle$, where $\|\alpha\|^2 + \|\beta\|^2 + \|\gamma\|^2 = 1$. Qutrits are manipulated in a similar manner to qubits; however, there are additional gates which may be performed on qutrits. We focus on the X_{+1} and X_{-1} operations, which are addition and subtraction gates, modulo 3. For example, X_{+1} elevates $|0\rangle$ to $|1\rangle$ and elevates $|1\rangle$ to $|2\rangle$, while wrapping $|2\rangle$ to $|0\rangle$.

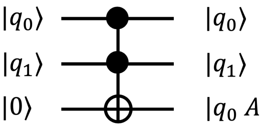


Fig. 11. ***Reversible AND circuit using a single ancilla bit. The inputs are on the left, and time flows rightward to the outputs. This AND gate is implemented using a Toffoli (CCNOT) gate with inputs q_0, q_1 and a single ancilla initialized to 0. At the end of the circuit, q_0 and q_1 are preserved, and the ancilla bit is set to 1 if and only if both other inputs are 1.***

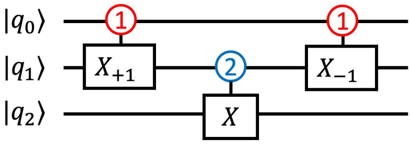


Fig. 12. ***Toffoli decomposition via qutrits. Each input and output is a qubit. The red controls activate on $|1\rangle$ and the blue controls activate on $|2\rangle$. The first gate temporarily elevates q_1 to $|2\rangle$ if both q_0 and q_1 were $|1\rangle$. We then perform the X operation only if q_1 is $|2\rangle$. The final gate restores q_0 and q_1 to their original state.***

Just as single-qubit gates have qutrit analogs, the same holds for two-qutrit gates. For example, consider the CNOT operation, where an X gate is performed conditioned on the control being in the $|1\rangle$ state. For qutrits, an X_{+1} or X_{-1} gate may be performed, conditioned on the control being in any of the three possible basis states. Just as qubit gates are extended to take multiple controls, qutrit gates are extended similarly.

In Fig. 12, a Toffoli decomposition using qutrits is given. A similar construction for the Toffoli gate is known from the past work [38], [55]. The goal is to perform an X operation on the last (target) input qubit q_2 if and only if the two control qubits, q_0 and q_1, are both $|1\rangle$. First, a $|1\rangle$-controlled X_{+1} is performed on q_0 and q_1. This elevates q_1 to $|2\rangle$ if and only if q_0 and q_1 were both $|1\rangle$. Then, a $|2\rangle$-controlled X gate is applied to q_2. Therefore, X is performed only when both q_0 and q_1 were $|1\rangle$, as desired. The controls are restored to their original states by a $|1\rangle$-controlled X_{-1} gate, which undoes the effect of the first gate. The key intuition in this decomposition is that the qutrit $|2\rangle$ state can be used instead of ancilla to store temporary information.

B. Generalized Toffoli Gate

The intuition of our technique extends to more complicated gates. In particular, we consider the generalized Toffoli gate, a ubiquitous quantum operation which extends the Toffoli gate to have any number of control inputs. The target input is flipped if and only if all of the controls are activated. Our qutrit-based circuit decomposition for the generalized Toffoli gate is presented in Fig. 13. The decomposition is expressed in terms of three-qutrit gates (two controls and one target) instead of single- and two- qutrit gates because the circuit can be understood purely classically at this granularity.

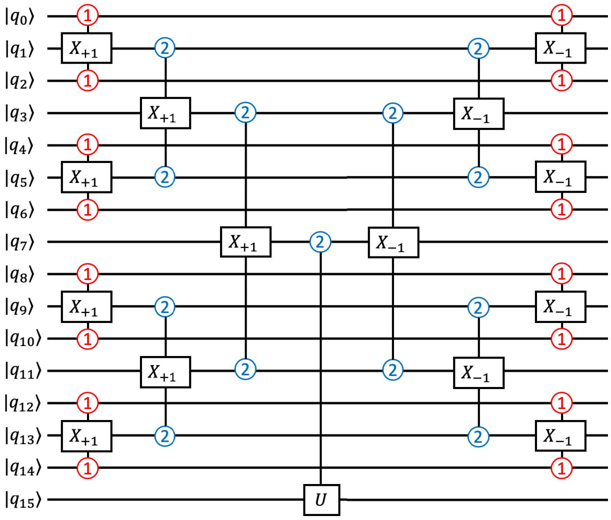


Fig. 13. ***Our circuit decomposition for the generalized Toffoli gate is shown for 15 controls and 1 target. The inputs and outputs are both qubits, but we allow occupation of the $|2\rangle$ qutrit state in between. The circuit has a tree structure and maintains the property that the root of each subtree can only be elevated to $|2\rangle$ if all of its control leaves were $|1\rangle$. Thus, the U gate is only executed if all controls are $|1\rangle$. The right-half of the circuit performs uncomputation to restore the controls to their original state. This construction applies more generally to any multiply controlled U gate. Note that the three-input gates are decomposed into six two-input and seven single-input gates in our actual simulation, as based on the decomposition in [17].***

In actual implementation and in our simulation, we used a decomposition [17] that requires six two-qutrit and seven single-qutrit physically implementable quantum gates.

Our circuit decomposition is most intuitively understood by treating the left half of the circuit as a tree. The desired property is that the root of the tree, q_7, is $|2\rangle$ if and only if each of the 15 controls was originally in the $|1\rangle$ state. To verify this property, we observe the root q_7 can only become $|2\rangle$ if and only ifq_7 was originally $|1\rangle$ and q_3 and q_{11} were both previously $|2\rangle$. At the next level of the tree, we see q_3 could have only been $|2\rangle$ if q_3 was originally $|1\rangle$ and both q_1 and q_5 were previously $|2\rangle$, and similarly for the other triplets. At the bottom level of the tree, the triplets are controlled on the $|1\rangle$ state, which are activated only when the even-index controls are all $|1\rangle$. Thus, if any of the controls were not $|1\rangle$, the $|2\rangle$ states would fail to propagate to the root of the tree. The right-half of the circuit performs uncomputation to restore the controls to their original state.

After each subsequent level of the tree structure, the number of qubits under consideration is reduced by a factor of $\sim$2. Thus, the circuit depth is logarithmic in N, which is exponentially smaller than ancilla-free qubit-only circuits. Moreover, each qutrit is operated on by a constant number of gates, so the total number of gates is linear in N.

Table 1 Scaling of Circuit Depths and Two-Qudit Gate Counts for All Three Benchmarked Circuit Constructions for the *N*-Controlled Generalized Toffoli

	QUBIT	QUBIT+ANCILLA	QUTRIT
Depth	$\sim 633N$	$\sim 76N$	$\sim 38\log_2(N)$
Gate Count	$\sim 397N$	$\sim 48N$	$\sim 6N$

We verified our circuits, both formally and via simulation. Our verification scripts are available on our GitHub [4].

C. Simulation Results

Table 1 shows the scaling of circuit depths and two-qudit gate counts for all three benchmarked circuits. The QUBIT-based circuit constructions from the past work are linear in depth and have a high linearity constant. Augmenting with a single borrowed ancilla (QUBIT+ANCILLA) reduces the circuit depth by a factor of 8. However, both circuit constructions are significantly outperformed by our QUTRIT construction, which scales logarithmically in N and has a relatively small leading coefficient. Although there is not an asymptotic scaling advantage for two-qudit gate count, the linearity constant for our QUTRIT circuit is 70× smaller than for the equivalent ancilla-free QUBIT circuit.

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

The full results of our circuit simulations are shown in Fig. 14. All simulations are for the 14-input (13 controls and 1 target) generalized Toffoli gate. We simulated each of the three circuit benchmarks against each of our noise models (when applicable), yielding the 16 bars in the figure. Note that our qutrit circuit consistently outperforms qubit circuits, with advantages ranging from 2× to 10 000×.

D. Discussion

The results presented in our work in [26] and [27] are applicable to QC in the near term, on machines that are expected within the next five years. By breaking the qubit abstraction barrier, we extend the frontier of what is computable by quantum hardware right now, without needing to wait for better hardware. As verified by our open-source circuit simulator coupled with realistic noise models, our circuits are more reliable than qubit-only equivalents, suggesting that qutrits offer a promising path toward scaling quantum computers. We propose further investigation into what advantage qutrits or qudits may confer. More broadly, it is critical for quantum architects to bear in mind that standard abstractions in classical computing do not necessarily transfer to quantum computation. Often,

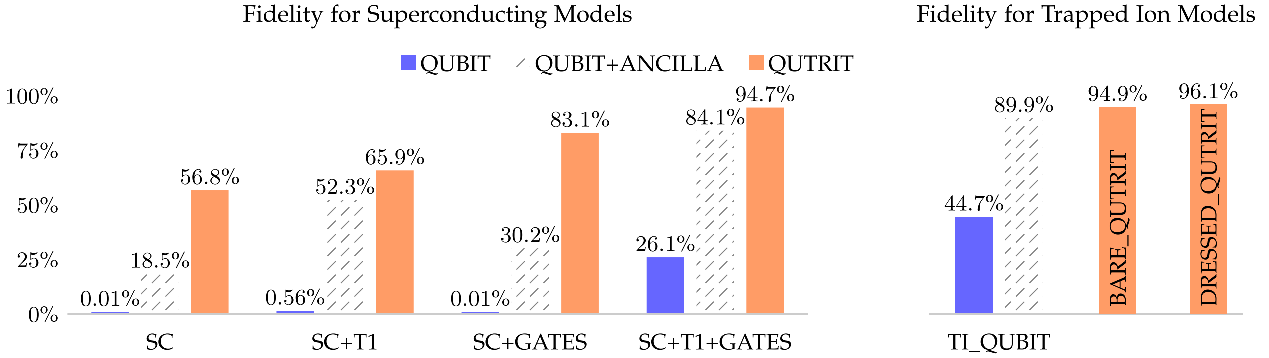


Fig. 14. ***Circuit simulation results for all possible pairs of circuit constructions and noise models. Each bar represents 1000+ trials, so the error bars are all $2\sigma < 0.1\%$. Our QUTRIT construction significantly outperforms the QUBIT construction. The QUBIT+ANCILLA bars are drawn with dashed lines to emphasize that it has access to an extra ancilla bit, unlike our construction. Figure reprinted with permission from [27].***

this presents unrealized opportunities, as in the case of qutrits.

V. BREAKING THE QUBIT ABSTRACTION VIA THE GKP ENCODING

Currently, there are many competing physical qubit implementations. For example, the transmon qubits [2] are encoded in the lowest two energy levels of the charge states in SC *LC* circuits with Josephson junctions; trapped ion qubits can be encoded in two ground state hyperfine levels [9] or a ground state level and an excited level of an ion [13]; quantum dot qubits use electron spin triplets [41]. These QIP platforms have rather distinct physical characteristics, but they are all exposed to the other layers in the stack as qubits and other implementation details are often hidden. This abstraction is natural for classical computing stack because the robustness of classical bits decouples the programming logic from physical properties of the transistors except the logical value. In contrast, qubits are fragile so there are more than (superpositions of) the logical values that we want to know about the implementation. For example, in the transmon qubits and trapped ion qubits, logical states can be transferred to higher levels of the physical space by unwanted operations and this can cause leakage errors [24], [71]. It will be useful for other layers in the stack to access this error information and develop methods to mitigate it. In Section IV, we discussed the qutrit approach that directly uses the third level for information processing, however, it could be more interesting if we can encode the qubit (qudit) using the whole physical Hilbert space to avoid leakage errors systematically and use the redundant degrees of freedom to reveal information about the noise in the encoding. The encoding proposed by Gottesman *et al.* [29] provides such an example. GKP encoding is free of leakage errors and other errors (in the form of small shifts in phase space) can be identified and corrected by quantum nondemolition (QND) measurements and simple linear optical operations. In realistic implementations of approximate GKP states (see Section V-C), there are leakage errors between logical states, but the transfer probability is estimated to be at the order of 10^{-10} with current techonology, thus negligible.

A. Phase Space Diagram

We describe the GKP qubits in the phase space. For a comparison, we first discuss the phase space diagram for a classical harmonic oscillator (CHO) and an SC qubit.

1) Classical Harmonic Oscillators: Examples of CHOs include *LC* circuits, springs, and pendulums with small displacement. The voltage/displacement (denoted as p) and the current/momentum (denoted as q) value completely characterize the dynamics of CHO systems. The phase space diagram plots p versus q, which for CHOs are circles (up to normalization) with the radius representing the system energy. The energy of CHOs can be any nonnegative real value.

2) Quantum Harmonic Oscillators: The QHO is the quantized version of the CHO and is the physical model for SC *LC* circuits and SC cavity modes. One of the values get quantized for QHOs is the system energy, which can only take equally spaced nonzero discrete values (see Fig. 16). The lowest allowed energy is not 0 but $(1/2)$ (up to normalization). We call the quantum state with the lowest energy the ground state. For a motion with a certain energy, the phase space diagram is not a circle anymore but a quasidistribution that can be described by the Wigner function. We say the distribution is a "quasi" distribution because the probability can be negative. The phase space diagram for the ground state and first excited state is plot in Fig. 15.

3) SC Charge Qubits: The QHO does not allow us selectively address the energy levels, thus leakage errors will occur if we use the lowest two levels as the qubit logic space. For example, a control signal that provides the energy difference ΔE enables the transition $|0\rangle \rightarrow |1\rangle$, but will also make the transition $|1\rangle \rightarrow |2\rangle$ which brings the state out of the logic space. To avoid this problem,

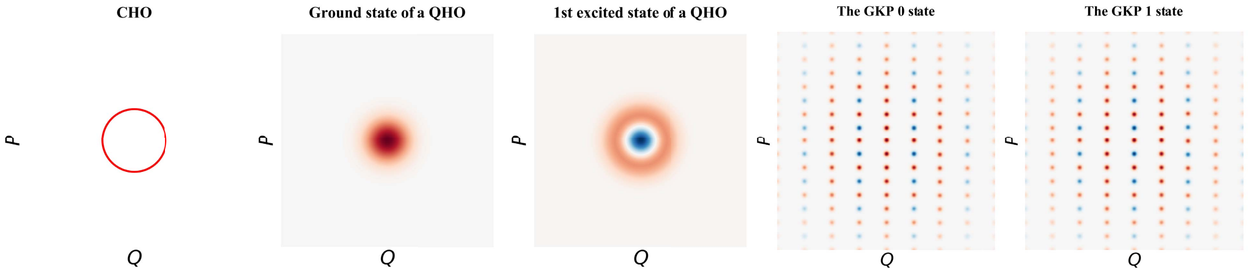


Fig. 15. ***Phase space diagrams for a CHO, the ground state, and the first excited state of a QHO and the logic 0 and 1 state of the GKP qubit. For quantum phase space diagrams, the plotted distribution is the Wigner quasi-probability function, where red indicates positive values and blue indicates negative values.***

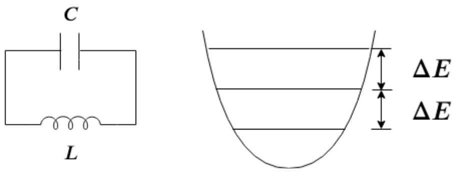


Fig. 16. ***Left: an LC circuit. In SC LC circuits, normal current becomes SC current. Right: the energy potential of a harmonic oscillator. In QHOs like the SC LC circuits, the system energy becomes equally spaced discrete values. The plotted two levels are the ground state and the first excited state.***

the Cooper pair box (CPB) design of an SC charge qubit replaces the inductor (see Fig. 17) with a Josephson junction, making the circuit an anharmonic oscillator, in which the energy levels are not equally spaced anymore. The Wigner function for CPB eigenstates are visually similar to those of QHO and only differ from them to the first order of the anharmonicity, thus we do not plot them in Fig. 15 separately.

B. Heisenberg Uncertainty Principle

We hope that with utilizing the whole physical states (higher energy levels), we can use the redundant space to encode and extract error information. However, the Heisenberg uncertainty principle sets the fundamental limit on what error information we can extract from the physical states—the more we know about the q variable, the less we know about the p variable. For example, we can “squeeze” the ground state of the QHO (also known as the vacuum state) in the p-direction; however, the distribution in the q-direction spreads, as shown in Fig. 18. Usually, we have to know both the p and q values to characterize the error information unless we know the error is biased. Thus, it is a great challenge to design encodings in the phase space to reveal error information.

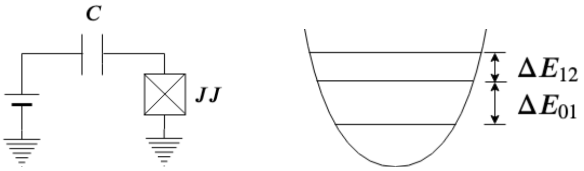


Fig. 17. ***Left: an LC circuit. In SC LC circuits, normal current becomes SC current. Right: the energy potential of a harmonic oscillator. In QHOs like the SC LC circuits, the system energy becomes equally spaced discrete values. The plotted two levels are the ground state and the first excited state.***

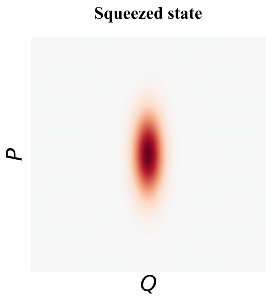


Fig. 18. ***Squeezed vacuum state.***

C. GKP Encoding

The GKP states are also called the grid states because each of them is a rectangular lattice in the phase space (see Fig. 15). There are also other types of lattice in the GKP family, for example, the hexagonal GKP [29]. Intuitively, the GKP encoding “breaks” the Heisenberg uncertainty principle—we do not know what are the measured p and q values of the state (thus expected values of p and q remain uncertain), but we do know that they must be integer multiples of the spacing of the grid. Thus, we have access to the error information in both directions and if we measure values that are not multiples of the spacing of the grid, we know there must be errors. Formally, the ideal GKP logical states are given by

$$\begin{aligned}|\bar{0}\rangle_{\text{gkp}} &= \sum_{k=-\infty}^{\infty} S_p^k |q=0\rangle \\ |\bar{1}\rangle_{\text{gkp}} &= \sum_{k=-\infty}^{\infty} S_p^k |q=\sqrt{\pi}\rangle \qquad (1)\end{aligned}$$

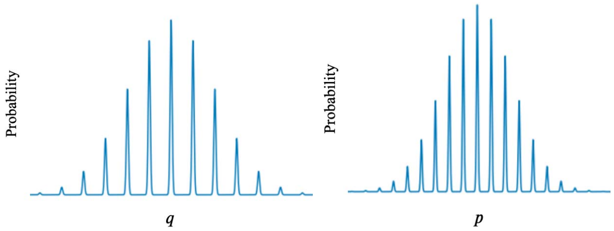


Fig. 19. ***Approximate GKP $|0\rangle$ state in q- and p-axis.***

where $S_p = e^{-2i(\pi)^{1/2}p}$ is the displacement operator in q space, which shifts a wave function in the q-direction by $2(\pi)^{1/2}$. These definitions show that for GKP logical 0 and 1, the spacing of the grid in q-direction is $2(\pi)^{1/2}$ and the spacing in p is $(\pi)^{1/2}$. In q-direction, the logical $|0\rangle$ state has peaks at even multiples of $(\pi)^{1/2}$, and the logical $|1\rangle$ state has peaks at odd multiples of $(\pi)^{1/2}$. For logical $|+\rangle$ and $|-\rangle$, the spacing in p and q grids is switched.

1) Approximate GKP States: The ideal GKP states require infinite energy, thus they are not realistic. In the laboratory, we can prepare approximate GKP states as illustrated in Fig. 19, where peaks and the envelope are Gaussian curve.

2) Error Correction With GKP Qubits: GKP qubits are designed to correct shift errors in q- and p-axis. A simple decoding strategy will be shifting the GKP state back to the closest peak. For example, if we measure a q value to be $2(\pi)^{1/2} + \Delta q$, where $\Delta q < ((\pi)^{1/2}/2)$, then we can shift it back to $2(\pi)^{1/2}$. With this simple decoding, GKP can correct all shift errors smaller than $((\pi)^{1/2}/2)$. While there are other proposals for encoding qubits in QHO [45], [50], [52] that are designed for realistic errors such as photon loss, it is shown that GKP qubits have the most error correcting ability in the regime of experimental relevance [5].

In addition, GKP qubits can also provide error correction information when concatenating with quantum error correction codes (QECCs) and yield higher thresholds. For example, when combining the GKP qubits with a surface code, the measured continuous p and q values in the stabilizer measurement can reveal more about the error distribution than traditional qubits [23], [51], [69].

Finally, it has been shown that given a supply of GKP-encoded Pauli eigenstates, universal fault-tolerant quantum computation can be achieved using only Gaussian operations [8]. Compared to qubit error correction codes, the GKP encoding enables much simpler fault-tolerant constructions.

D. Fault-Tolerant Preparation of Approximate GKP States

The GKP encoding has straightforward logical operation and promising error correcting performance. However, the difficulty of using GKP qubits in QIP platforms lies in its preparation since they live in highly nonclassical states with relatively high mean photon number (i.e., the average energy levels). Thus, reliable preparation of encoded GKP states is an important problem. In [64], we gave fault-tolerance definitions for GKP preparation in SC cavities and designed a protocol that fault-tolerantly prepares the GKP states. We briefly describe the main ideas.

1) Goodness of 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 $((\pi)^{1/2}/2)$ with certainty. For example, suppose we have an approximate $|\tilde{0}\rangle$ GKP state with a peak at $q = 0$ subject to a shift error e^{-ivp} with $|v| \leq ((\pi)^{1/2}/2)$. The finite width of the Gaussian peaks will have a nonzero overlap in the region $((\pi)^{1/2}/2) < q < (3(\pi)^{1/2}/2)$ and $(-3(\pi)^{1/2}/2) < q < (-(\pi)^{1/2}/2)$. Thus, with nonzero probability the state can be decoded to $|\tilde{1}\rangle$ instead of $|\tilde{0}\rangle$ (see Fig. 20 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.

2) Preparation of Approximate GKP States Using Phase Estimation: We observe that the GKP states are the eigenstates of the S_p operator, thus we can use phase estimation to gradually project a squeezed vacuum state to an approximate GKP state. The phase estimation circuit for preparing an approximate $|\tilde{0}\rangle$ GKP state is given in Fig. 21. The first horizontal line represents the cavity mode that we want to prepare the GKP states. The second line is a

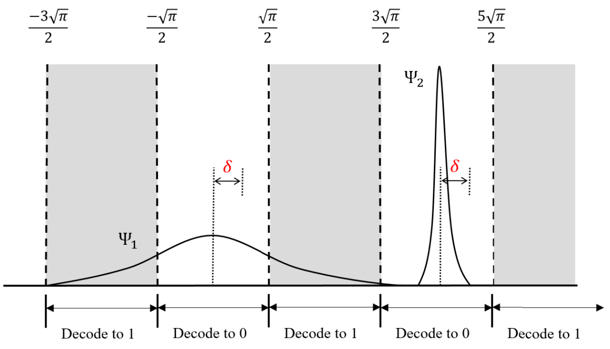


Fig. 20. ***Peaks centered at even integer multiples of $(\pi)^{1/2}$ in q space. The peak on the left contains large tails that extend into the region where a shift error is decoded to the logical $|\tilde{1}\rangle$ state. The peak on the right is much narrower. Consequently for some interval δ, the peak on the right will correct shift errors of size $((\pi)^{1/2}/2) - \delta$ with higher probability than the peak on the left.***

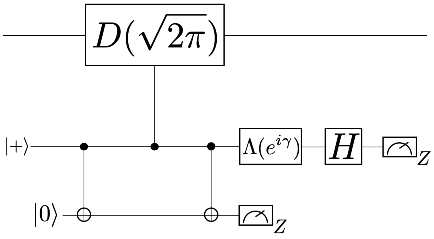


Fig. 21. ***Phase estimation circuit with the flag qubit. The protocol is aborted if the flag qubit measurement is nontrivial.***

transmon ancilla initialized to $|+\rangle$. The third line is a transmon flag qubit initialized to $|0\rangle$. The H gate is the Hadamard gate. $\Lambda(e^{i\gamma}) = \text{diag}(1, e^{i\gamma})$ is the gate with a control parameter γ in each round of the phase estimation to increase the probability of projecting the cavity state to an approximate eigenstate of the displacement operator after the measurement. After applying several rounds of the circuit in Fig. 21, the input squeezed vacuum state is projected onto an approximate eigenstate of S_p with some random eigenvalue $e^{i\theta}$. 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.

In our protocol, we use a flag qubit to detect any damping event during the controlled-displacement gate, if a nontrivial measurement is obtained, we abort the protocol and start over. Using our simulation results, we also find a subset of output states that are robust to measurement errors in the transmon ancilla and only accept states in that subset. We proved that our protocol is fault-tolerant according to the definition we gave. In practice, our protocol produces "good" approximate GKP states with high probability and we expect to see experimental efforts to implement our protocol.

E. Discussion

The GKP qubit architecture is a promising candidate for both near-term and fault-tolerant QC implementations. With intrinsic error-correcting capabilities, the GKP qubit breaks the abstraction layer between error correction and the physical implementation of qubits. In [64], we discussed the fault-tolerant preparation of GKP qubits and realistic experimental difficulties. We believe that qubit encodings like the GKP encoding will be useful for reliable QC.

VI. CONCLUSION AND FUTURE DIRECTIONS

In this review, we proposed that greater quantum efficiency can be achieved by breaking abstraction layers in the QC stack. We examined some of the previous work in this direction that are closing the gap between current quantum technology and real-world QC applications. We would also like to briefly discuss some promising future directions along this line.

A. 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 crosstalk. Emergent error modes are hard to characterize and to mitigate. Recently, it has been shown that randomized compiling could transform complicated noise channels including crosstalk, SPAM errors, and readout errors into simple stochastic Pauli errors [70], 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.

B. Algorithm-Level Error Correction

Near-term quantum algorithms such as variational quantum eigensolver (VQE) and 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) [63] and the Majorana loop stabilizer code (MLSC) [30] 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 works 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.

C. 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 [42] and error correction [31] suggests 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 QC stack. ■

REFERENCES

[1] *Cramming More Power Into a Quantum Device*. Accessed: Aug. 30, 2010. [Online]. Available: https://www.ibm.com/blogs/research/2019/03/power-quantum-device/

[2] J. Koch, "Charge-insensitive qubit design derived from the Cooper pair box," *Phys. Rev. A, Gen. Phys.*, vol. 76, no. 4, Oct. 2007, Art. no. 042319.

[3] (2018). *Cirq: A Python Framework for Creating, Editing, and Invoking Noisy Intermediate Scale Quantum (NISQ) Circuits*. [Online]. Available: https://github.com/quantumlib/Cirq

[4] (2019). *Code for Asymptotic Improvements to Quantum Circuits Via Qutrits*. [Online]. Available: https://github.com/epiqc/qutrits

[5] V. V. Albert *et al.*, "Performance and structure of single-mode bosonic codes," *Phys. Rev. A, Gen. Phys.*, vol. 97, no. 3, p. 32346, Mar. 2018.

[6] G. Aleksandrowicz *et al.*, "Qiskit: An open-source framework for quantum computing," IBM T.J Watson Res. Center, New York, NY, USA, Tech. Rep., 2019, doi: 10.5281/zenodo.2562111.

[7] F. Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[8] B. Q. Baragiola, G. Pantaleoni, R. N. Alexander, A. Karanjai, and N. C. Menicucci, "All-Gaussian universality and fault tolerance with the Gottesman-Kitaev-Preskill code," 2019, *arXiv:1903.00012*. [Online]. Available: http://arxiv.org/abs/1903.00012

[9] B. B. Blinov, D. Leibfried, C. Monroe, and D. J. Wineland, "Quantum computing with trapped ion hyperfine qubits," in *Experimental Aspects of Quantum Computing*. New York, NY, USA: Springer, 2005, pp. 45–59.

[10] S. A. Caldwell *et al.*, "Parametrically activated entangling gates using transmon qubits," *Phys. Rev. A, Gen. Phys. Appl.*, vol. 10, no. 3, Sep. 2018, Art. no. 034050.

[11] A. M. Childs, E. Schoute, and C. M. Unsal, "Circuit transformations for quantum architectures," Tech. Rep., 2019, doi: 10.4230/LIPIcs.TQC.2019.3.

[12] J. M. Chow *et al.*, "Simple all-microwave entangling gate for fixed-frequency superconducting qubits," *Phys. Rev. Lett.*, vol. 107, no. 8, Aug. 2011, Art. no. 080502.

[13] J. I. Cirac and P. Zoller, "Quantum computations with cold trapped ions," *Phys. Rev. Lett.*, vol. 74, no. 20, pp. 4091–4094, May 1995.

[14] W. A. Cross, S. L. Bishop, A. J. Smolin, and M. J. Gambetta, "Open quantum assembly language," 2017, *arXiv:1707.03429*. [Online]. Available: https://arxiv.org/abs/1707.03429

[15] P. de Fouquieres, S. G. Schirmer, S. J. Glaser, and I. Kuprov, "Second order gradient ascent pulse engineering," *J. Magn. Reson.*, vol. 212, no. 2, pp. 412–417, Oct. 2011.

[16] 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*, vol. 536, no. 7614, pp. 63–66, Aug. 2016.

[17] Y.-M. Di and H.-R. Wei, "Elementary gates for ternary quantum logic circuit," 2011, *arXiv:1105.5485*. [Online]. Available: http://arxiv.org/abs/1105.5485

[18] A. Erhard *et al.*, "Characterizing large-scale quantum computers via cycle benchmarking," *Nature Commun.*, vol. 10, no. 1, p. 5347, Nov. 2019, doi: 10.1038/s41467-019-13068-7.

[19] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014, *arXiv:1411.4028*. [Online]. Available: http://arxiv.org/abs/1411.4028

[20] X. Fu *et al.*, "eQASM: An executable quantum instruction set architecture," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, 2019, pp. 224–237.

[21] X. Fu *et al.*, "An experimental microarchitecture for a superconducting quantum processor," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2017, pp. 813–825.

[22] X. Fu *et al.*, "A microarchitecture for a superconducting quantum processor," *IEEE Micro*, vol. 38, no. 3, pp. 40–47, May 2018.

[23] K. Fukui, A. Tomita, A. Okamoto, and K. Fujii, "High-threshold fault-tolerant quantum computation with analog quantum error correction," *Phys. Rev. X*, vol. 8, no. 2, May 2018, Art. no. 021054.

[24] J. Ghosh, A. G. Fowler, J. M. Martinis, and M. R. Geller, "Understanding the effects of leakage in superconducting quantum-error-detection circuits," *Phys. Rev. A, Gen. Phys.*, vol. 88, no. 6, Dec. 2013, Art. no. 062329.

[25] S. J. Glaser, U. Boscain, T. Calarco, C. P. Koch, W. Köckenberger, R. Kosloff, I. Kuprov, B. Luy, S. Schirmer, T. Schulte-Herbrüggen, D. Sugny, and F. K. Wilhelm, "Training Schrödinger's cat: Quantum optimal control," *Eur. Phys. J. D*, vol. 69, no. 12, p. 279, Dec. 2015.

[26] P. Gokhale, J. M. Baker, C. Duckering, F. T. Chong, N. C. Brown, and K. R. Brown, "Extending the frontier of quantum computers with qutrits," *IEEE Micro*, vol. 40, no. 3, pp. 64–72, 2020.

[27] P. Gokhale, J. M. Baker, C. Duckering, N. C. Brown, K. R. Brown, and F. T. Chong, "Asymptotic improvements to quantum circuits via qutrits," in *Proc. 46th Int. Symp. Comput. Architectural*, New York, NY, USA, Jun. 2019, pp. 554–566, doi: 10.1145/3307650.3322253.

[28] P. Gokhale *et al.*, "Partial compilation of variational algorithms for noisy intermediate-scale quantum machines," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchitecture*, New York, NY, USA, Oct. 2019, pp. 266–278.

[29] D. Gottesman, A. Kitaev, and J. Preskill, "Encoding a qubit in an oscillator," *Phys. Rev. A, Gen. Phys.*, vol. 64, no. 1, p. 12310, Jun. 2001.

[30] Z. Jiang, J. McClean, R. Babbush, and H. Neven, "Majorana loop stabilizer codes for error mitigation in fermionic quantum simulations," *Phys. Rev. Appl.*, vol. 12, no. 6, pp. 064041-1–064041-17, Dec. 2019. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevApplied.12.064041, doi: 10.1103/PhysRevApplied.12.064041.

[31] E. Kapit, "Hardware-efficient and fully autonomous quantum error correction in superconducting circuits," *Phys. Rev. Lett.*, vol. 116, no. 15, Apr. 2016, Art. no. 150501.

[32] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 359–392, Jan. 1998.

[33] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S. J. Glaser, "Optimal control of coupled spin dynamics: Design of NMR pulse sequences by gradient ascent algorithms," *J. Magn. Reson.*, vol. 172, no. 2, pp. 296–305, Feb. 2005.

[34] A. Y. Kitaev, A. H. Shen, and M. N. Vyalyi, *Classical and Quantum Computation*. Providence, RI, USA: AMS, 2002.

[35] P. V. Klimov *et al.*, "Fluctuations of energy-relaxation times in superconducting qubits," *Phys. Rev. Lett.*, vol. 121, Aug. 2018, Art. no. 090502.

[36] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, "A quantum engineer's guide to superconducting qubits," *Appl. Phys. Rev.*, vol. 6, no. 2, Jun. 2019, Art. no. 021318.

[37] B. P. Lanyon *et al.*, "Towards quantum chemistry on a quantum computer," *Nature Chem.*, vol. 2, no. 2, pp. 106–111, Feb. 2010.

[38] B. P. Lanyon *et al.*, "Simplifying quantum logic using higher-dimensional Hilbert spaces," *Nature Phys.*, vol. 5, no. 2, pp. 134–140, Feb. 2009.

[39] N. Leung, M. Abdelhafez, J. Koch, and D. Schuster, "Speedup for quantum optimal control from automatic differentiation based on graphics processing units," *Phys. Rev. A, Gen. Phys.*, vol. 95, no. 4, Apr. 2017, Art. no. 042318.

[40] N. M. Linke *et al.*, "Experimental comparison of two quantum computing architectures," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3305–3310, Mar. 2017.

[41] D. Loss and D. P. DiVincenzo, "Quantum computation with quantum dots," *Phys. Rev. A, Gen. Phys.*, vol. 57, no. 1, pp. 120–126, Jan. 1998.

[42] Y. Lu *et al.*, "Universal stabilization of a parametrically coupled qubit," *Phys. Rev. Lett.*, vol. 119, no. 15, Oct. 2017.

[43] D. Maslov, S. M. Falconer, and M. Mosca, "Quantum circuit placement," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 4, pp. 752–763, Apr. 2008.

[44] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," *New J. Phys.*, vol. 18, no. 2, 2016, Art. no. 023023.

[45] M. H. Michael *et al.*, "New class of quantum error-correcting codes for a bosonic mode," *Phys. Rev. X*, vol. 6, no. 3, Jul. 2016, Art. no. 031006.

[46] P. Murali, J. M. Baker, A. Javadi-Abhari, F. T. Chong, and M. Martonosi, "Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst. (ASPLOS)*, New York, NY, USA, Apr. 2019, pp. 1015–1029.

[47] P. Murali, N. M. Linke, M. Martonosi, A. J. Abhari, N. H. Nguyen, and C. H. Alderete, "Full-stack, real-system quantum computer studies: Architectural comparisons and design insights," in *Proc. 46th Int. Symp. Comput. Architectural (ISCA)*, New York, NY, USA, Jun. 2019, pp. 527–540.

[48] P. Murali, D. C. Mckay, M. Martonosi, and A. Javadi-Abhari, "Software mitigation of crosstalk on noisy intermediate-scale quantum computers," in *Proc. 25th Int. Conf. Architectural Support Program. Lang. Operating Syst. (ASPLOS)*, New York, NY, USA, Mar. 2020, pp. 1001–1016.

[49] A. M. Nielsen and L. I. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th ed. New York, NY, USA: Cambridge Univ. Press, 2011.

[50] 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 Trans. Inf. Theory*, vol. 65, no. 4, pp. 2563–2582, Apr. 2019.

[51] K. Noh and C. Chamberland, "Fault-tolerant bosonic quantum error correction with the surface-Gottesman-Kitaev-Preskill code," *Phys. Rev. A, Gen. Phys.*, vol. 101, no. 1, Jan. 2020, Art. no. 012316.

[52] N. Ofek *et al.*, "Demonstrating quantum error correction that extends the lifetime of quantum information," 2016, *arXiv:1602.04768*. [Online]. Available: http://arxiv.org/abs/1602.04768

[53] A. Pavlidis and E. Floratos, "Arithmetic circuits for multilevel qudits based on quantum Fourier transform," 2017, *arXiv:1707.08834*. [Online]. Available: http://arxiv.org/abs/1707.08834

[54] Qiskit. (2019). *Qiskit NoiseAdaptiveLayout Pass*. Accessed: Aug. 1, 2019. [Online]. Available: https://github.com/Qiskit/qiskit-terra/pull/2089

[55] T. C. Ralph, K. J. Resch, and A. Gilchrist, "Efficient Toffoli gates using qudits," *Phys. Rev. A, Gen. Phys.*, vol. 75, no. 2, Feb. 2007, Art. no. 022313.

[56] J. Randall, A. M. Lawrence, S. C. Webster, S. Weidt, N. V. Vitanov, and W. K. Hensinger, "Generation of high-fidelity quantum control methods for multilevel systems," *Phys. Rev. A, Gen. Phys.*, vol. 98, no. 4, Oct. 2018, Art. no. 043414.

[57] J. Randall *et al.*, "Efficient preparation and detection of microwave dressed-state qubits and qutrits with trapped ions," *Phys. Rev. A, Gen. Phys.*, vol. 91, no. 1, Jan. 2015, Art. no. 012322.

[58] Rigetti. (2019). *PyQuil*. Accessed: Aug. 1, 2019. [Online]. Available: https://github.com/rigetticomputing/pyquil

[59] R. Quilc. (2019). *Use Swap Fidelity Instead of Gate Time as a Distance Metric*. Accessed: Aug. 1, 2019. [Online]. Available: https://github.com/rigetti/quilc/pull/395

[60] M. Sarovar, T. Proctor, K. Rudinger, K. Young, E. Nielsen, and R. Blume-Kohout, "Detecting crosstalk errors in quantum information processors," 2019, *arXiv:1908.09855*. [Online]. Available: https://arxiv.org/abs/1908.09855

[61] N. Schuch and J. Siewert, "Natural two-qubit gate for quantum computation using theXYinteraction," *Phys. Rev. A, Gen. Phys.*, vol. 67, no. 3, Mar. 2003, Art. no. 032301.

[62] 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," 2007, *arXiv:0712.3227*. [Online]. Available: http://arxiv.org/abs/0712.3227

[63] K. Setia, S. Bravyi, A. Mezzacapo, and J. D. Whitfield, "Superfast encodings for fermionic quantum simulation," *Phys. Rev. Res.*, vol. 1, no. 3, pp. 033033-1–033033-8, Oct. 2019. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevResearch.1.033033, doi: 10.1103/PhysRevResearch.1.033033.

[64] Y. Shi, C. Chamberland, and A. Cross, "Fault-tolerant preparation of approximate GKP states," *New J. Phys.*, vol. 21, no. 9, 2019, Art. no. 093007.

[65] Y. Shi *et al.*, "Optimized compilation of aggregated instructions for realistic quantum computers," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst. (ASPLOS)*, New York, NY, USA, Apr. 2019, pp. 1031–1044, doi: 10.1145/3297858.3304018.

[66] *Strawberry Fields*, Xanadu, Toronto, ON, Canada, 2016.

[67] S. S. Tannu and M. Qureshi, "Ensemble of diverse mappings: Improving reliability of quantum computers by orchestrating dissimilar mistakes," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, New York, NY, USA, Oct. 2019, pp. 253–265.

[68] S. S. Tannu and M. K. Qureshi, "Not all qubits are created equal: A case for variability-aware policies for NISQ-era quantum computers," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst. (ASPLOS)*, New York, NY, USA, Apr. 2019, pp. 987–999.

[69] C. Vuillot, H. Asasi, Y. Wang, L. P. Pryadko, and B. M. Terhal, "Quantum error correction with the toric Gottesman-Kitaev-Preskill code," *Phys. Rev. A, Gen. Phys.*, vol. 99, no. 3, Mar. 2019, Art. no. 032344.

[70] J. J. Wallman and J. Emerson, "Noise tailoring for scalable quantum computation via randomized compiling," *Phys. Rev. A, Gen. Phys.*, vol. 94, no. 5, Nov. 2016, Art. no. 052325.

[71] C. J. Wood and J. M. Gambetta, "Quantification and characterization of leakage errors," *Phys. Rev. A, Gen. Phys.*, vol. 97, no. 3, Mar. 2018.

[72] A. Zulehner, A. Paler, and R. Wille, "An efficient methodology for mapping quantum circuits to the IBM QX architectures," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 38, no. 7, pp. 1226–1236, 2019.

[73] E. Pednault, J. A. Gunnels, G. Nannicini, L. Horesh, and R. Wisnieff, "Leveraging secondary storage to simulate deep 54-qubit sycamore circuits," 2019, *arXiv:1910.09534*. [Online]. Available: https://arxiv.org/abs/1910.09534

[74] C. Huang *et al.*, "Classical simulation of quantum supremacy circuits," 2020, *arXiv:2005.06787*. [Online]. Available: https://arxiv.org/abs/2005.06787

[75] M. Kjaergaard *et al.*, "Superconducting qubits: Current state of play," *Annu. Rev. Condens. Matter Phys.*, vol. 11, no. 1, pp. 369–395, 2020, doi: 10.1146/annurev-conmatphys-031119-050605.