Superstaq:

Deep Optimization of Quantum Programs

Colin Campbell*, Frederic T. Chong*, Denny Dahl*, Paige Frederick*, Palash Goiporia*, Pranav Gokhale*, Benjamin Hall*, Salahedeen Issa*, Eric Jones*, Stephanie Lee*, Andrew Litteken*, Victory Omole*, David Owusu-Antwi*, Michael A. Perlin*, Rich Rines*, Kaitlin N. Smith*, Noah Goss†, Akel Hashim†, Ravi Naik†, Ed Younis‡, Daniel Lobser§, Christopher G. Yale§, Benchen Huang¶, Ji Liu∥*Infleqtion (these authors contributed equally)

[†]Quantum Nanoelectronics Laboratory, University of California at Berkeley [‡]Computational Research Division, Lawrence Berkeley National Laboratory

§Sandia National Laboratories
¶University of Chicago

||Argonne National Laboratory

Abstract—We describe Superstaq, a quantum software platform that optimizes the execution of quantum programs by tailoring to underlying hardware primitives. For benchmarks such as the Bernstein-Vazirani algorithm and the Qubit Coupled Cluster chemistry method, we find that deep optimization can improve program execution performance by at least 10x compared to prevailing state-of-the-art compilers. To highlight the versatility of our approach, we present results from several hardware platforms: superconducting qubits (AQT @ LBNL, IBM Quantum, Rigetti), trapped ions (QSCOUT), and neutral atoms (Infleqtion). Across all platforms, we demonstrate new levels of performance and new capabilities that are enabled by deeper integration between quantum programs and the device physics of hardware.

Index Terms-Quantum compilation, cross-layer optimization

I. INTRODUCTION

Despite sustained progress in quantum hardware, there is a substantial gap to utility-scale quantum computation. On one hand, there has been consistent improvement in gate fidelities over the past decade. For instance, since the advent of superconducting transmon qubits [1], two-qubit gate errors have been lowered by $\sim 0.77\times$ per year [2]. We see similar rates of progress in other qubit technologies including neutral atoms [3], [4] and trapped ions [5], [6]. While this progress is encouraging, hardware progress alone would require at

This material is supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research under Award Number DE-SC0021526 and under Ouantum Testbed Program Contracts No. DE-AC02-05CH11231 and DE-NA0003525. This material is based upon work supported by the U.S. Department of Energy, Office of Science, National Quantum Information Science Research Centers. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. We acknowledge the use of IBM Quantum services for this work. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team.

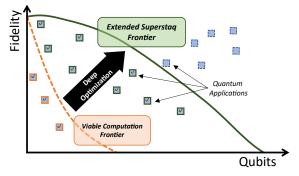


Fig. 1: Deep optimization advances the frontier of which quantum programs can be run successfully.

least a decade to achieve societally-useful outcomes such as simulating molecules relevant to fertilizer production [7].

We view quantum software as a force multiplier that can significantly shorten the timeline for utility-scale results from quantum hardware. There are compelling parallels to classical computing: the world's top computing facilities bolster their state-of-the-art hardware capabilities with significant investment in software tools such as CUDA [8], Jax [9], OpenMP [10], and SLURM [11]. Similarly, software tooling—especially for compilation—can enable users to extract better results from quantum hardware, both for near-term systems as well as for upcoming large-scale fault-tolerant computers.

In fact, we find even stronger motivation for optimized compilation in the quantum setting than in the classical setting. First, quantum resources are far more expensive than classical resources. For instance, an error-corrected quantum NAND gate is expected to be 10 *billion* times more expensive (\sim 10 qubitseconds vs. 10^{-9} transistorseconds) than a classical NAND gate [12]. Second, for foreseeable quantum computers, optimized compilation will be *necessary* to bring useful applications within the boundary of achievable computations. Lastly, applications brought within this boundary must exhibit exponential or high-degree polynomial quantum

TABLE I: Superstag hardware targets in this paper.

Platform	Qubit Type	Techniques Highlighted			
AQT @Berkeley	Supercond. (fixed-freq. transmon)	approx. synthesis, BYOG, PMW-4, qutrit decomposition			
IBM	Supercond. (fixed-freq. transmon)	pulse cancellation, dynamical decoupling, fractional gates			
Infleqtion Hilbert	Neutral Atom (Cesium)	global gate decomposition and scheduling			
Rigetti Aspen M-3	Supercond. (modu- lated transmon)	dynamical decoupling			
QSCOUT @Sandia	Trapped ion (171 Yb ⁺)	SWAP Mirroring, fractional gates			

advantage [12], that can immediately justify high compilation costs. Thus, investment in deep compiler optimization can enable applications that are otherwise out of reach for current hardware at various scales, as depicted in Fig. 1.

In this spirit and with this motivation, we present Superstag, a quantum software platform that invokes deep optimization by compiling quantum programs to low levels of underlying hardware control to extract as much performance as possible. We emphasize that while compilation to standard textbook gatesets (including work on optimal qubit mapping and routing [13], [14]) is addressed by existing techniques, such as open-source tools like Qiskit [15] and theoretical frameworks like the KAK decomposition [16] & Solovay-Kitaev Theorem [17], optimized compilation to lower levels remains a relatively underexplored territory. With Superstag, we emphasize a full-stack approach to improving quantum compilation across multiple quantum technologies. We demonstrate that cross-layer (i.e., breaking abstractions) and device-physics-aware optimization significantly improves the performance of algorithms on quantum hardware, enabling functionality that is years ahead of the baseline hardware frontier.

We have evaluated our optimized compilation techniques on multiple qubit technologies, in each case, compiling to the lowest level of hardware primitives accessible. In this paper, we present results from five platforms, shown in Tab. I, that showcase Superstaq's deep cross-layer optimization. The remainder of this paper describes the highest-leverage compiler optimizations, with experimental validation for each. Section II presents results that emerge from optimization of quantum programs to lower abstraction levels (native gateset or pulses) than typically considered. Section III presents Superstaq's integration of dynamical decoupling, which we find to be a particularly profitable technique for mitigating noise. Finally Section IV presents the *star-to-line* routing, a simple-but-effective technique that bridges the gap from low-connectivity hardware to typical applications.

Our software can be accessed through open-source Python frontends, cirq-superstaq and qiskit-superstaq, available through PyPI via pip install. Both packages include comprehensive libraries of custom gates css.ops.qubit_gates, css.ops.qudit_gates, and qss.custom_gates, that support Superstaq's

interaction with the low-level hardware primitives referenced in this paper.

II. OPTIMIZED DECOMPOSITION

Superstaq's approach to optimized decomposition begins by defining the native gateset of the hardware. We define a device's "native gates" to be the smallest operation that has a known expected unitary. For example, on many hardware platforms, multi-qubit gates are implemented by an echoed interaction, whereby interaction Hamiltonians are applied in two half-steps: positive and negative directions. The isolated half-steps do not have a known expected unitary (due to an uncharacterized error term), but the composite sequence does. We therefore consider the full echoed sequence as the level of abstraction that forms our native gateset. We expand on this with the concrete example of Echoed Cross-Resonance gates on IBM hardware in the next subsection.

By first identifying the native gateset of each device, Superstaq is able to tailor circuit decompositions to exploit the full capabilities of the hardware. It combines a number of novel, platform-aware decomposition and optimization strategies, as well as off-the-shelf optimizers available in packages like Cirq, Qiskit, and BQSKit [18]. In future iterations we also plan to incorporate *superoptimization* passes [19] to further ensure the optimal translation to a given gateset.

This section explores a few of the specific optimized decomposition strategies employed by Superstaq for a variety of hardware platforms. In each case we find that this cross-layer approach extends the capabilities of the device beyond what would be possible with a hardware-agnostic approach.

A. Superconducting Gateset on IBM (Echoed Cross-Resonance)

We begin by describing Superstaq's optimizations for IBM superconducting hardware. At a high level, our compiler performs three key steps: (a) qubit mapping, (b) decomposing into native gatesets to surface opportunities for cross-gate pulse cancellation, and then (c) merging single-qubit gates to minimize rotation angle and prefer use of virtual R_z rotations over physical R_x rotations.

On current IBM hardware, two-qubit interactions are performed with the cross-resonance pulse [20], with Hamiltonian:

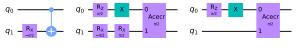
$$\hat{H} = \omega_{ix} \frac{\hat{I}\hat{X}}{2} + \omega_{iz} \frac{\hat{I}\hat{Z}}{2} + \omega_{zi} \frac{\hat{Z}\hat{I}}{2} + \omega_{zx} \frac{\hat{Z}\hat{X}}{2} + \omega_{zz} \frac{\hat{Z}\hat{Z}}{2}.$$

The cross-resonance interaction amounts to driving the control qubit at the natural frequency of the target qubit. On IBM devices, the ω_{zx} frequency is used to calibrate an entangling gate. By applying this Hamiltonian in an echoed fashion with both positive and negative drives, with an X gate in between, the resulting unitary implements only the $\hat{Z}\hat{X}$ interaction, plus a side effect due to the X gate to reduce error [21]. This is the native two-qubit gate we target with Superstag.

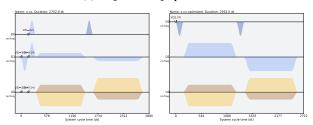
We demonstrate Superstaq optimizing a Qubit-Coupled Cluster circuit [22], [23], useful for quantum chemistry applications. The core subcircuit of QCC consists of an R_z

operation sandwiched by four total CX gates, two pairs each staggered in a "ladder" formation on either side [24], as seen in Fig. 3a. Superstag compiles this circuit using (a) cross-gate pulse cancellations [25] and (b) as-late-as-possible scheduling. Firstly, Superstag leverages cross-gate pulse cancellations by decomposing the CX gate into single-qubit gates and an echoed cross-resonance. Decomposing the CX into its native gates reveals further single-qubit gate decompositions that are unavailable to Qiskit, even at its highest optimization level.

Before describing the full optimization of the QCC circuit, first we'll describe a representative warm-up example involving an R_x gate followed by a CX in Fig. 2. Qiskit will execute this circuit directly. Superstaq decomposes the CX into native gates: (1) an $R_z(\pi/2)$ gate followed by an X gate on the first qubit, (2) an $R_x(\pi/2)$ on the second qubit, and (3) an echoed cross-resonance gate interacting the first and second qubits (Fig. 2a) [26]. Superstaq then cancels the $R_x(\pi/2)$ gate with the preceding $R_x(-\pi/2)$ gate on the second qubit, reducing the duration of the resulting pulse schedule executed on hardware (Fig. 2b).



(a) Superstaq optimization.



(b) Comparison of pulse schedules generated with Qiskit (optimization_level=3) and Superstaq.

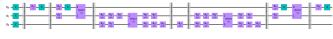
Fig. 2: (a) A circuit consisting of a X-rotation followed by a CX gate (left), along with the subsequent decomposition into native gates (center) and cross-gate cancellation carried out by Superstaq (right). (b) The Qiskit schedule of the original circuit (left, 2752dt ≈ 612 ns) and the Superstaq optimized schedule (right, 2592dt ≈ 576 ns), achieving a savings of 160dt ≈ 36 ns, exactly the duration of a single-qubit gate. (Note: the amplitude scale for each channel is normalized.)

Now we again consider the core QCC subcircuit. We first decompose the CXs in the subcircuit to obtain the result shown in Fig. 3b. In this initial decomposition, Superstag reveals a number of single-qubit gate cancellation opportunities similar to as seen in the warm-up example. In particular we see instances of the following gate sequences:

- X-Rz (π) -X-Rz $(\pi/2)$ -X on q_0
- X-Rx $(\pi/2)$ on q_1
- Rz $(\pi/2)$ -Rx $(\pi/2)$ -Rz $(\pi/2)$ -Rz $(\pi/2)$ -Rz $(\pi/2)$ on q_1
- Rz $(\pi/2)$ -Rx $(\pi/2)$ -Rx $(\pi/2)$ on q_1
- X-Rz $(\pi/2)$ -Rx $(\pi/2)$ on q_2
- Rz $(\pi/2)$ -Rx $(\pi/2)$ -Rz $(3\pi/2)$ -Rz $(\pi/3)$ -Rz $(\pi/2)$ -Rx $(\pi/2)$ on q_2



(a) Exemplar core subcircuit for QCC



(b) Decomposed core QCC subcircuit



(c) Superstaq optimized core QCC subcircuit

Fig. 3: Superstaq optimization of the Qubit Coupled Cluster ansatz (QCC).

Each of these sequences can be recombined into either (a) shorter sequences consisting of R_z and R_x gates, or (b) sequences that reduce the use of R_x gates in favor of R_z gates. In addition to the reduction in number of gates with (a), because R_z gates on IBM devices are virtual [27], (b) achieves additional savings. The resultant re-compiled sequences are, respectively:

- Rz $(3\pi/2)$ -X on q_0
- Rz (π) -Rx $(\pi/2)$ -Rz (π) on q_1
- Rz (π) -Rx $(\pi/2)$ -Rz (π) on q_1
- Rz $(\pi/2)$ -X on q_1
- Rz $(\pi/2)$ -Rx $(\pi/2)$ -Rz $(\pi/2)$ on q_2
- Rz $(3\pi/2)$ -Rx $(\pi/2)$ -Rz $(5\pi/3)$ -Rx $(\pi/2)$ -Rz (π) on q_2

The final optimized result is shown in Fig. 3c. We achieve a savings of $320 \text{dt} \approx 71 \text{ns}$ in pulse duration for the QCC subcircuit. As shown in Fig. 4, we executed the Superstaq-optimized QCC circuits on IBM quantum hardware. Superstaq increases Hellinger fidelity by 2.3 x and 13 x compared to Qiskit's highest optimization, level-3.

This is a promising start, but we can go even further. Note that the cross-resonance implements a $\mathcal{ZX}(\pi/2)$, but with pulse-stretching, Superstag provides an $AceCR(\theta)$ gate implementing a parametrized $\mathcal{ZX}(\theta)$. The parametric cross-resonance gives users the ability to compile to $\mathcal{ZX}(\theta)$ directly in hardware, rather than decompose it into a native gate sequence. For example, the parametric cross-resonance can replace the standard cross-resonance in a CX implementation, turning it into a fractional gate $(CX)^{\alpha}$ for $\alpha = \theta/2\pi$ that results in shorter pulse sequences that are more accurate to evaluate, thus allowing faster convergence for variational algorithms [28]–[30].

We now consider the Quantum Approximate Optimization Algorithm (QAOA), a variational quantum algorithm used to solve combinatorial optimization [31]. The QAOA ansatz consists of three stages: (1) a layer of Hadamards putting each qubit into an equal superposition, (2) a ZZ-type interaction between desired pairs of qubits based on the problem Hamiltonian, and (3) a final mixing layer of R_x rotation gates. In general, this ansatz can be extended to larger depth for

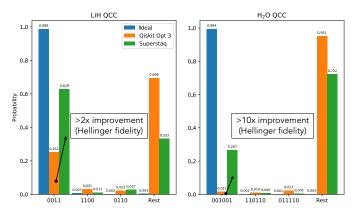


Fig. 4: Qubit Coupled Cluster experimental results on ibm_lagos device for LiH (4 qubits, 31 two-qubit gates) and H₂O (6 qubits, 57 two-qubit gates), 32k shots per bar. Superstaq improves Hellinger fidelity $28.6\% \rightarrow 66.3\%$ (2.3x) and $2.1\% \rightarrow 28.2\%$ (13x) respectively. Ideal results are almost (but not completely) single-peaked.

more accurate convergence by repeating the final two stages in sequence p times. We consider a two-qubit toy model of the p = 1 QAOA ansatz, as shown in the left hand side of Fig. 5a, consisting of the standard decomposition for a single ZZ-type interaction. Out of the box, Superstaq optimizes this decomposition via cross-gate pulse cancellations between each CX gate and the surrounding single-qubit gates, similar to in Figs. 2 and 3. Additionally, Superstag's aforementioned isolation of the cross-resonance at the gate-level allows implementing entangling gates other than the ZX-type interaction via phase kickback, of the form $\{\sigma_i \sigma_i \mid i, j \in x, y, z\}$ for Pauli operators σ_i (see Figure 4 in [32]). For example, the ZZ interaction in Fig. 5a can be implemented directly with the $AceCR(\theta)$, as shown in the right hand side of Fig. 5a, requiring fewer two-qubit native gates. We compare the gate errors (1 - Hellinger fidelities) for Qiskit, Superstaq's standard optimization, and the direct $AceCR(\theta)$ implementation facilitated by Superstaq in Fig. 5b and we see that except for $\gamma = 7\pi/4$, Superstag either ties or beats Qiskit i.e. the orange bar is tied with or lower than the blue bar. The error bar is the standard error of the mean $\frac{\sigma}{\sqrt{N}}$ for N trials, and a tie occurs when the error bars intersect. We note that while Qiskit beats the direct $AceCR(\theta)$ implementation for $\gamma \in \{0.25\pi, 0.75\pi, \pi\}$, future iterations of Superstag will address this with custom pulse-shaping of the CR, rather than relying on Qiskit's calibration.

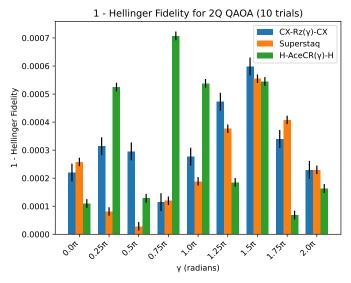
B. Neutral Atom Gateset

Here we discuss Superstaq's compilation to Inflequion's Hilbert QPU, which has a native gateset comprising:

- Two-qubit CZ gates.
- Single-qubit rotations of the form $R_z(\theta) = e^{-i\theta Z_j/2}$, where Z_j is the Pauli-Z operator for qubit j.
- Global rotations of the form $GR_{\phi}(\theta) = e^{-\mathrm{i}\theta S_{\phi}}$, where the spin operator $S_{\phi} = \frac{1}{2} \sum_{j} (\cos(\phi) X_{j} + \sin(\phi) Y_{j})$



(a) Two-qubit QAOA ansatz.



(b) 1 - Hellinger fidelity results for $\gamma \in [0, 2\pi]$ and $\beta = \pi/2$ fixed.

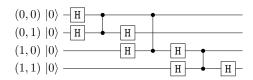
Fig. 5: (a) Qiskit ZZ decomposition (two CX gates) vs. direct (i.e., with AceCR) ZZ decomposition. (b) Plots comparing error between Qiskit ZZ decomposition, standard Superstaq, and Superstaq AceCR for $\gamma \in [0, 2\pi]$.

generates homogeneous rotations of all qubits in the X-Y plane, and X_i, Y_i are Pauli operators for qubit j.

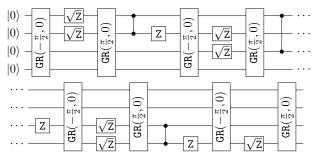
The error budget of a quantum computation is typically dominated by two-qubit gates (e.g., CZ gates), and the neutral atom architecture is no exception. Leveraging a large volume of literature on minimizing two-qubit gate costs, the Superstag compiler first uses decompositions available in Cirq [33] to convert an arbitrary circuit into a gateset of CZ gates and arbitrary single-qubit gates. However, Hilbert's gateset has the nonstandard feature that a global GR gate addressing all qubits is in turn required to implement an arbitrary single-qubit gate. Superstag's Hilbert compiler therefore optimizes to the capabilities and limitations of the GR gate.

While CZ and R_z gates on the neutral atom architecture occur on time scales measured in hundreds of nanoseconds, the $GR_\phi(\theta)$ gate takes a time that is directly proportional to its *pulse area* θ , and takes a few microseconds when $\theta=\pi$. Minimizing the usage of GR gates therefore directly translates into decreased circuit runtimes in this neutral atom architecture.

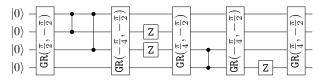
When decomposing arbitrary single-qubit gates into GR and R_z gates, it is possible to "recycle" GR gates and decompose single-qubit gates in parallel. The Superstag compiler therefore has two basic strategies for minimizing GR gate usage: (a) scheduling single-qubit gates in such a way as



(a) A GHZ circuit for four qubits at the corners of a square.



(b) A straightforward compilation of the GHZ circuit for Hilbert.



(c) The same circuit compiled using Superstag.

Fig. 6: (a) A GHZ circuit for four qubits at the corners of a square, decomposed down to CZ gates and single-qubit Hadamards. (b) A straightforward decomposition of the GHZ circuit, in which each Hadamard is decomposed as $H=e^{-i\pi/4}Z\cdot GR_x(\frac{\pi}{2})\cdot \sqrt{Z}\cdot GR_x(-\frac{\pi}{2})$, and pairs of Hadamards are decomposed using the same GR gates. (c) The same circuit decomposed using Superstag, which optimizes single-gate parallelization and minimizes GR gate usage.

to minimize the number of times that a collection of single-qubit gates must be decomposed into GR and R_z gates, and (b) decomposing any given collection of single-qubit gates in such a way as to minimize the required pulse area of GR gates in the decomposition.

Single-qubit gates are scheduled by a greedy algorithm that iterates over the operations in a circuit in topological order to collect maximal sets of single-qubit gates that can be executed in parallel [34]. This algorithm is both efficient and optimal in reducing the number of single-qubit gate collections that must be decomposed into global gates.

Ignoring global phases, each single-qubit gate can be written in the form $Z^{z+a}X^xZ^{-a}$ with $x\in[0,1]$, where if x=0 the gate can simply be merged into the next single-qubit gate with $x\neq 0$ on the same qubit. The minimum global pulse area required to implement such a gate is $x\pi$, and the minimum global pulse area required to implement a collection $\mathcal C$ of such gates in parallel is $\max_{j\in\mathcal C} x_j\pi$. Superstag achieves this minimum with a decomposition of the gates in $\mathcal C$ into local R_z gates and two GR gates that are mutual inverses, and each have a pulse area of $\min_{j\in\mathcal C} x_j\pi/2$. The phase ϕ of the GR gates in this decomposition is chosen in such a way as

to minimize the number of R_z gates in the compiled circuit [34].

As a final point, we note that the assignment of single-qubit gates to a minimal collection of parallelizable gates is an over-determined problem. After an initial scheduling pass and prior to decomposing single-qubit gates, Superstag therefore additionally assigns single-qubit gates to collections in such a away as to minimize the net GR pulse area of their decompositions.

Fig. 6 shows an example of equivalent circuits for preparing a four-qubit GHZ state, compiled either using off-the-shelf methods in Cirq (see figure caption), or using Superstaq. In this example, the techniques in Superstag reduce the R_z gate count from 10 to 3, the GR gate count from 8 to 5, and the net GR pulse area from 4π to 1.5π . Using an experimentally motivated [35] depolarizing noise model with a single-qubit SPAM fidelity of 0.98, GR fidelity of 0.999 (per qubit), R_z fidelity of 0.99, and a CZ fidelity of 0.96, we estimate that the circuits in Fig. 6 should prepare a four-qubit GHZ state with respective fidelities of 0.71 (straightforward compilation) and 0.78 (Superstag compilation). Remarkably, this backof-the-envelope estimate is in quantitative agreement with our experimental trials on Hilbert, which use population measurements and parity oscillations to extract GHZ fidelities of 0.726(9) and 0.782(8). This agreement should not to be taken too seriously due to the simplicity of the noise model – R_z and CZ gates are known to be dominated by phase errors, for example. Nonetheless, as a rough comparison we find that the same improvement to the GHZ fidelity of the straightforwardly compiled circuit can be achieved by improving the CZ fidelity from 0.96 to 0.99. While this comparison should not be taken literally, it illustrates the benefit of an improved compiler, which complements hardware improvements. Note that quantum errors compound multiplicatively throughout a circuit, such that the benefits of an improved compiler grow with increasing circuit size.

C. Trapped Ion Gateset

We next describe Superstag optimization for QSCOUT [36], the trapped ion testbed at Sandia Lab. QSCOUT's gateset is generated via Raman transitions of a hyperfine 'clock' transition of ¹⁷¹Yb⁺ ions and consists of continuously parameterized single-qubit rotations, $R_{\phi}(\theta)$ about any axis in the X/Y plane, a virtual $Z(\theta)$ gate, and a continuously parameterized Mølmer-Sørensen $(\mathcal{MS}_{\phi}(\theta))$ gate.

1) Entangling Basis Compilation: The $\mathcal{MS}_{\phi}(\theta)$ interaction is an XX-type interaction of the form $XX(\theta) = e^{-i\frac{\theta}{2}\sigma_X\otimes\sigma_X}$, and we adjust θ via the amplitude of one of the Raman beams [37]. However, the bare interaction, $MS_{\phi}^{cu}(\theta)$, requires Raman transitions generated by counter-propagating beams for motional sensitivity, while the single-qubit gates use a single beam with two Raman tones, a co-propagating configuration. To mitigate phase instabilities that occur when mixing gates of differing propagation, a bare $MS_{xx}^{cu}(\theta)$ is sandwiched first by counter-propagating rotation gates, $R_y^{cu}(\pm \pi/2)$, converting the interaction into a ZZ-type interaction [38], $\mathcal{ZZ}(\theta)$, and

then a set of co-propagating rotation gates $R^{co}_{\phi+\pi/2}(\pm\pi/2)$, converting it back into an XX-type interaction, $\mathcal{MS}_{\phi}(\theta)$, in which the phase ϕ is encoded in the co-propagating rotation gates. Superstaq uses just the internal $\mathcal{ZZ}(\theta)$ portion of the MS gate when decomposing arbitrary two-qubit operations, requiring fewer single-qubit operations than would the standard compilation.

This is shown below, where,

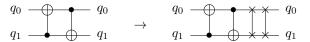
$$\mathcal{MS}_{\phi}(heta)$$

is equal to,

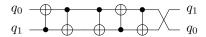
As identified above, the internal portion of the circuit corresponds to a $ZZ(\theta)$ interaction. This means that making $ZZ(\theta)$ the compiler's native entangling operation instead of $\mathcal{MS}_{\phi}(\theta)$ provides opportunities to cancel some single-qubit gates. The standard decomposition of an arbitrary two-qubit gate due to the KAK decomposition is,

where each two-qubit gate can be implemented with a single $\mathcal{MS}(\theta)$ or $\mathcal{ZZ}(\theta)$ application and surrounding single-qubit gates. However, by compiling to \mathcal{ZZ} as the native entangling operation, the co-propagating gates originally required for each \mathcal{MS} will instead be merged with one another or with the outer single-qubit unitaries $\{A,B,C,D\}$ to decrease the overall depth of the circuit.

2) SWAP mirroring: SWAP Mirroring is built on the observation that there are instances when appending two SWAPs to an arbitrary two-qubit gate can be more efficient than performing the gate by itself. For example, transforming a double-CX circuit by appending two SWAPs (identity):



The last SWAP can be performed "virtually" by relabeling the qubits and the other SWAP can be decomposed into the CX basis:



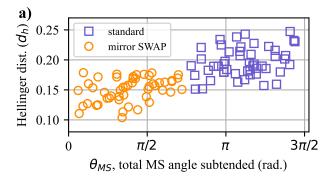
The CX gates cancel twice to result in:

$$q_0 \longrightarrow q_1$$

Thus, a two-operation circuit is optimized to a single operation.

SWAP mirroring is described more generally by leveraging Equation B9 in Appendix B of [39] which is described by the circuit:

Given an arbitrary two-qubit unitary, we use the KAK decomposition (1) and compare it to decomposing via the above circuit. We determine which is the more efficient circuit first by whichever version yields a reduction in the number of Mølmer-Sørensen (MS) gates (both $\mathcal{MS}_{\phi}(\theta)$ and $\mathcal{ZZ}(\theta)$), and then by the least total MS angle subtended across all MS gates, θ_{MS} , within the decomposition. Applying SWAP mirroring to Haar random 2-qubit unitaries yields 38 percent less θ_{MS} when successful and 17 percent average reduction in θ_{MS} in general circuits overall. This demonstration is relevant for applications like Supercheq fingerprinting [40] and quantum volume [39].



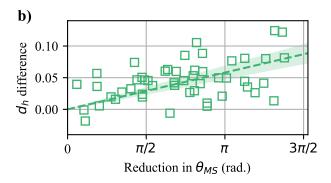


Fig. 7: Experimental results for SWAP mirroring in the context of FSim(Θ , Φ) gates on the QSCOUT trapped ion hardware. a) For fifty randomly chosen Θ and Φ , the Hellinger distance, d_h , between the empirically observed and simulated ideal values for the output probability distribution is plotted versus θ_{MS} , for both the standard compilation (blue) and the SWAP mirrored compilation (orange). b) For each FSim implementation, the difference between d_h of the standard and SWAP mirrored compilations is plotted against the total reduction θ_{MS} . Linear fit (dashed) with 2σ error bars emphasizes trend of increased d_h difference with increased θ_{MS} reduction.

On QSCOUT, we investigate SWAP mirroring as a means to reduce θ_{MS} , within a circuit in order to improve its performance. For this demonstration, we use the $FSim(\Theta, \Phi)$ gate and restrict ourselves to instances where $\pi/4 < \Theta < 3\pi/4$ and $\pi/2 < \Phi < 3\pi/2$. This range of parameter inputs of the FSim provides a region in which SWAP mirroring always provides a reduction in θ_{MS} . We randomly choose 50 different variations of FSim parameters Θ and Φ , and compile each FSim gate via the standard methodology or invoking SWAP mirroring. The circuit compilations consist of a series of $R_{\phi}(\theta)$ gates, virtual $Z(\theta)$ gates, and three $\mathcal{Z}\mathcal{Z}(\theta)$ gates. In each case, we begin with an input state of $|10\rangle$, and measure the output state. The Hellinger distance, d_h of the output state is computed in relation to the ideal output state of the circuit. In Fig.7a, we plot the d_h of the output states of both the standard and SWAP mirrored compilation against the total MS angle subtended by the circuit, showing a clear improvement in the performance of the circuit under SWAP mirroring conditions. In Fig.7b, we look directly at the difference of d_h per circuit and plot as a function of the reduction in θ_{MS} , also showing a trend for greater differences at increased θ_{MS} reductions.

D. "Bring-Your-Own" Gateset (BYOG)

Any benefit from optimizing decompositions to a device's native gateset is ultimately limited by how well those native gates themselves can be implemented in the hardware. That is, the performance of the optimized circuit is critically dependent on the precise calibration of hardware operations to implement the gateset assumed by the compiler. This accuracy is fundamentally limited by the precision and stability of the device and control hardware, as well as the budget available for calibration (which can be a source of considerable experimental overhead, as device and control system properties can be vary qubit-to-qubit and day-to-day). High-fidelity circuit execution therefore often relies on techniques such as echo sequences (discussed above), composite gates, and optimal control to improve native gate precision, at the cost of increased circuit complexity and runtime.

We therefore employ an alternative paradigm. Instead of compiling to an ideal gateset and then requiring it to be reproduced as well as possible in hardware, we first find a set of quantum operations which is easily implemented in the hardware, and then pass those to the compiler to use when decomposing logical gates. The task of gate calibration is then in part reduced to that of characterization. Spatial and temporal inhomogeneity is handled naturally: it can just be the case that the native gateset might include (for example) slightly different entangling operations on different pairs of qubits, all of which may be different than they were yesterday. The experimenter can focus on calibrating out unwanted factors (such as crosstalk [41] to spectator qubits or leakage out of the logical subspace).

This compilation strategy comes at the cost of expressivity. It is generally the case that the uncalibrated gates will not generate arbitrary logical operations as efficiently as more conventional choices, which are typically chosen in part for

this expressiveness. This complexity can be reduced in part by employing approximate synthesis, in which logical operations are decomposed to sequences of native operations which are only required to approximate the desired unitary to within a provided tolerance (typically chosen to be just small enough that that approximation error does not contribute substantially to the overall error rate on the device). We then find that the loss in expressivity is often overcome by the reduction in complexity of the native operations themselves: while the same logical operations require a shorter sequence of native operations given a more conventional gateset, the physical implementation of each of these native operations is typically more complex due to the extended or composite sequences which are necessary to precisely implement the chosen native operation. An example of this tradeoff will be shown in Section II-D2.

1) Implementation: We implement this strategy for two-qubit operations on the Advanced Quantum Testbed (AQT), a superconducting transmon quantum computer at Lawrence Berkeley National Laboratory. Superstaq's compilation endpoint for AQT allows arbitrary unitary matrices to be assigned to pulse calibrations for the control hardware. Superstaq will then assume the provided gate definitions to be the device's native gateset when compiling logical operations into physical pulse sequences.

We leverage the Berkeley Quantum Synthesis Toolkit (BQSKit) [18] for approximate synthesis. BQSKit provides powerful tools for fast synthesis given arbitrary basis operations. After merging adjacent two-qubit logical operations into a single unitary, we employ its "QSearchSynthesis" pass [42] to approximate the unitary to a provided precision using a sequence of two-qubit native operations and arbitrary single-qubit gates. The tolerance is chosen to balance approximation fidelity with that lost due to increasing sequence lengths.

Single-qubit operations are then decomposed analytically into AQT's native single-qubit $R_x(\pi/2)$ gates and virtual-Z rotations. The decomposition chosen depends on the form of the provided unitary: if the virtual-Z rotation on that qubit can be commuted through the subsequent two-qubit operation we use the ubiquitous ZXZXZ decomposition; otherwise, we employ the PMW-4 sequence introduced in [43] to ensure that the virtual phase is returned to zero beforehand. In both cases, we employ optimizations like those described in [44] to decrease the number of $R_x(\pi/2)$ pulse required for certain subsets of single-qubit unitaries (e.g. a Hadamard gate requires just one pulse in the former case and three in the latter).

2) Extension to Qutrits: An immediate advantage to the BYOG model described above is that it is easily extended to higher-level systems by allowing for custom gate definitions of arbitrary dimension. Moving beyond the two-level (qubit) subspace is a significant way to expand the computational capability of a quantum device. By incorporating higher energy levels of the quantum systems, not only do we unlock an exponentially larger Hilbert space in which to perform quantum computations, but also higher effective connectivity than an equivalent system implemented on qubits. These factors

have been shown to offer asymptotic speedups for important operations [45]–[47] given access to three-level (qutrit) operations. The low anharmonicity of superconducting transmon qubits make them an ideal architecture for implementing qutrit gates, and indeed high-fidelity one- and two-qutrit quantum operations have already been demonstrated on AQT [48].

The BQSKit approximate synthesis pass we employ supports qutrit gates out of the box. We decompose single-qutrit gates analytically, assuming a fixed single-qutrit gateset comprising $R_x(\pi/2)$ gates acting in the $\{|0\rangle, |1\rangle\}$ and $\{|1\rangle, |2\rangle\}$ subspace of SU(3), and arbitrary virtual-Z rotations in either subspace. As in [49], arbitrary unitaries are first decomposed into a sequence three SU(2) operations in alternating twolevel subspaces of SU(3). Each subspace operation is then deconstructed via the ubiquitous ZXZXZ decomposition, using the available native operations acting in that subspace. The latter step allows for the same optimizations employed when decomposing single-qubit gates, in which each subspace operation may be implemented with zero or one $R_x(\pi/2)$ pulses if its rotation angle is sufficiently close to $0 \ (\pm \pi/2)$ (where "sufficiently close" can be configured to ensure a desired precision).

Finally, we note that this compilation strategy is compatible with Equivalent Circuit Averaging (ECA), another compilation technique implemented in Superstaq's compilation endpoint for AQT to mitigate systematic errors (first introduced in [44]). When using the BYOG compilation strategy, Superstaq's ECA compilation endpoint exploits the stochasticity of the approximate synthesis pass to generate an ensemble of logically equivalent but physically distinct decompositions of each gate, which in turn are used to generate the set of equivalent circuits to average over.

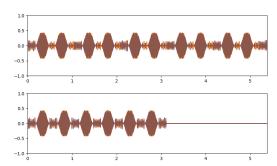


Fig. 8: Pulse sequences implementing the same randomly-generated SU(9) operation on AQT hardware, decomposed using a conventional qutrit-CZ native operation (top) and the measured unitary of the cross-Kerr pulse alone (bottom). Though the latter requires a longer gate decomposition (with six applications of its native two-qutrit gate instead of five), the physical pulse sequence is shorter because it forgoes the echoing required to implement a conventional qutrit-CZ gate natively.

3) Demonstration: To demonstrate the efficiency of the BYOG compilation strategy, in Fig. 8 we present two different Superstag-generated pulse sequences implementing the

same randomly-generated SU(9) operation using two different native entangling gates. The first uses the precise qutrit-CZ operation described in [48], which employs an echo sequence consisting of two cross-Kerr entangling pulses interleaved with X gates in the $\{|0\rangle, |1\rangle\}$ subspace to annihilate unwanted entangling phases. In the second, we provide Superstag with just the pulse sequence and measured unitary of the cross-Kerr pulse itself. In both cases the tolerance used for approximate synthesis was set to $5 \cdot 10^{-4}$ (Hilbert-Schmidt norm, or equivalently a process fidelity of 10^{-3}). We find that the qutrit-CZ is indeed more expressive, requiring just five qutrit-CZ operations to approximate the desired unitary compared to six two-qutrit operations when decomposed to the cross-Kerr pulse itself. However, because each gutrit-CZ gate itself comprises a sequence of two cross-Kerr interactions, this expressivity does not translate into a shorter physical implementation. As seen in Fig. 8, the sequence length is reduced by about 42% when we compile to the lower-level operation.

Finally, we validate this compilation procedure experimentally by using it to demonstrate a two-qutrit SWAP operation. The SWAP circuit (including setup and measurement operations) is compiled using Superstag's public-facing API (interfaced via cirg-superstag), to which we provide the pulse definitions and measured unitary of the aforementioned cross-Kerr interaction and pulse definitions for the relevant single-qubit native operations. The pulse sequences returned by Superstag's compilation endpoint for AQT are then implemented directly on the AQT hardware. Readout correction is applied by measuring and inverting the confusion matrices (cf. [50]). The resulting measurement frequencies for all computational-basis input and output states are shown in Fig. 9, from which we find an overall truth-table fidelity of about 73.5%. Note we do not employ ECA for this demonstration.

100>	-0.63	0.06	0.05	0.02		0.04	0.08	0.09	0.03-
01>	-0.01	0.01		0.7	0.01	0.09	0.04	0.1	0.04
02>	-0.05	0.04	0.02	-0.1	0.04	0.03	0.84	0.05	0.03
10>	0.07	0.69	0.02	0.03		0.02	0.03	0.1	0.03
11>	0.01	0.01	0.03		0.81	0.03	0.06	0.03	0.03
12>	-0.08	0.03	0.14	0.07	-0.03	0.03	0.04	0.56	0.07
20>	0.04	0.03	0.65	0.01	0.04	0.02	0.05	0.12	0.04
$ 21\rangle$	0.02	0.02	0.01		0.03	0.83	0.03	0.03	0.02
22>	0.02	0.01	0.01		0.01	-0.01	0.03	0.03	0.9
	00>	01>	02>	10>	11>	$ 12\rangle$	20>	21>	22>

Fig. 9: Truth table measurement frequencies of a qutrit-SWAP operation (after readout correction). The overall truth-table fidelity was approximately 73.5%.

III. DYNAMICAL DECOUPLING

A. Introduction

Noisy hardware is an important consideration when building a quantum compiler. Through cross layer optimization, Superstaq attempts to create performance gains at every layer of the stack. To this end, one of the most important and promising low-level techniques for quantum error suppression is known as dynamical decoupling (DD) [51], [52]. At a high level, DD works by injecting additional operations that are engineered to suppress the buildup of coherent errors in a circuit. In a noiseless setting, the operations injected by DD would cancel each other out and resolve to the identity. For this reason, a compiler designed merely to simplify and shorten circuits would eliminate DD operations, resulting in worse overall performance. Superstaq balances the objectives of shortening circuits with the benefits of DD, and has built a DD suite to target a variety of hardware backends.

The idea behind DD is that coupling between qubits and their environment can lead to undesired qubit evolution, such as stray rotation by an unknown angle on the Bloch sphere. These processes destroy the quantum information stored in a qubit. DD addresses these errors by applying regular pulse sequences that can be thought of as changing the "frame" in which the qubit stores information, e.g. by regularly swapping the north and south poles of the Bloch sphere. In the co-rotating frame of the qubit, the unknown environmental couplings then average out to zero, thereby eliminating the buildup of coherent error.

The simplest DD sequence is known as CPMG [53], [54], which simply applies periodic, evenly-spaced X gates, known as π -pulses, to a qubit. However, this sequence does not protect against stray R_x rotations, which commute with (and are therefore unaffected by) the CPMG pulses. This limitation can be fixed simply by alternating the axis of rotation X and Y, resulting in the so-called XY4 sequence (because four rotations are necessary for a single "period" that brings the qubit back to its original state). XY4 is thus the simplest universal DD sequence, which is to say that it can mitigate stray qubit rotations about any axis. Making further modifications to XY4, for example by using additional axes of rotation, recursively nesting DD sequences into the gaps between DD pulses, or using unevenly spaced DD pulses, yields a large family of DD sequences that exhibit different advantages in the presence of different environmental or control errors [52]. In all cases, DD works best when a compiler has control over the timing of the pulses that it injects into a circuit.

As a simple demonstration of the benefits of DD, Fig. 10 shows the results of an experiment to measure the relaxation (T_1) time of an idling qubit on Rigetti's Aspen M-3 quantum processor. We first initialize a qubit in the $|1\rangle$ state, and show the probability that the qubit is still observed in $|1\rangle$ upon measurement at a later time. We either let the qubit idle between state preparation and measurement, or we insert two evenly-spaced XY4 repetitions (8 pulses total) during the

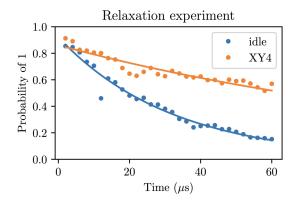


Fig. 10: After preparing a qubit in the state $|1\rangle$, interactions with the environment cause the qubit to lose energy and decay to $|0\rangle$ as it idles. However, inserting an XY4 DD sequence mitigates this process and extends the qubit lifetime. Lines show a fit to an exponential decay curve. Figure reproduced with permission from Ref. [55].

idle time. Altogether, inserting this DD sequence increased the lifetime of a qubit by a factor of 4.

B. Execution

An effective DD framework must have the ability to employ both (a) the diverse set of DD sequences referenced above and (b) various levels of concatenation during a period of qubit idling. Further, DD must have context of both program and machine properties during runtime for error mitigation to be most effective [56]. For example, many factors influence the noise a qubit encounters as it idles during program execution, causing some implementations of DD to be more effective than others. Exemplar factors include: duration of qubit idle windows (how many DD sequences can we fit and is there optimal spacing between the DD operations?); circuit information (what state is the idling qubit holding?); qubit coherence properties (will one type of DD correction be better than another for a qubit?); operator error rates (must balance tradeoffs between decoherence resilience and extra gate-induced error that each DD operation injects); parallel gate executions (does crosstalk affect an idling qubit?) native gate-set (what DD sequence is easiest to implement with built-in operations?); and architectural constraints of classical control hardware (how precise is the classical control infrastructure?). These factors simultaneously influence the optimal type of DD, in terms of operator sequence chosen and number of iterations within an idle window, that provides a quantum program with the greatest noise resilience during execution.

Quantum hardware with low level access typically provides a mapping of instructions to sequences of low-level control signals (pulses) that carry out the target instruction. The Superstaq DD optimizer uses this mapping to schedule circuits. In particular, when a circuit is scheduled, every circuit operation is annotated with its execution start time and duration. This timing data allows us to compute important circuit

information such as a circuit's critical paths and periods of qubit idling. For each qubit in an operation, we use the timing data to determine the start and end times of the qubit's idling periods. These idling durations are the points in the circuit that we target with appropriately spaced DD sequences, such as CPMG (XX), XY4, and XY8 [57], using individual and concatenated pulse sequences. Our implementation is written in a generalizable fashion that can flexibly accommodate other DD sequences as well.

We also take quantum computing hardware timing constraints into consideration when scheduling a circuit. If a circuit does not satisfy the system's timing criteria in terms of thresholds for pulse duration, alignment, and granularity, the scheduled circuit will not be executable because the circuit instructions cannot be realized with the low-level control hardware. An example of low-level control hardware would be the arbitrary waveform generators (AWGs) that drive superconducting transmon qubits (i.e. IBM's qubit technology). IBM's quantum hardware currently requires that pulses begin at a time that is a 16-fold multiple of a specified (by the device's timing constraints) alignment value expressed in device-dependent timescale increment, dt [58]. To respect this constraint, when compiling circuits targeting IBM's hardware we align each operation i to begin at $16k_i$ dts for a whole number k_i .

C. Results

Fig. 11 shows how DD makes algorithm execution more noise-robust. In this case study, we ran 4000 shots of a four-qubit implementation of the Bernstein-Vazirani (BV) benchmark (secret string = '1111') on the 27-qubit IBM Hanoi device. Fig. 11 compares a level-3 Qiskit optimization and Superstaq DD optimization to the ideal single-peak distribution. We see that invoking Superstaq DD improves the fidelity to the ideal distribution from 37% to 89%. Not only did Superstaq DD boost the likelihood of observing the correct all-1's outcome for the BV application, Superstaq DD made a previously unobtainable distribution feasible on the same hardware.

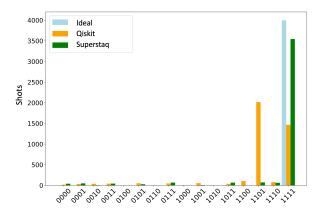


Fig. 11: DD results on IBM Hanoi quantum computer for 4-qubit Bernstein-Vazirani circuit with ('1111' key).

The Superstag DD optimizer was designed to encompass the "write once target all" objective of Superstag. It complements other circuit optimization passes, such as gatecount minimization and noise aware mapping, to produce DD scheduling that is customized for an algorithm and QC paring. In Fig. 12, Superstaq DD is compared to Qiskit's highest optimization level for five different machines using the fourqubit BV benchmark. To better understand how a circuit optimization mitigates noise and sharpens a distribution, the figure of merit known as relative strength [59] is often used. Relative strength for a distribution is defined as the ratio of total correct observations to most frequent incorrect observations. Superstag DD significantly improves program outcomes, with results as high as 68x on the Hanoi machine relative to the Qiskit baseline. Although Superstag DD had about the same relative strength as Oiskit on Mumbai and Toronto, we still observed improvements in the probability of success through Superstag.

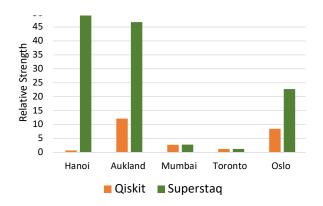


Fig. 12: DD results across five IBM machines for a four-qubit Bernstein-Vazirani program. Superstaq increases relative strength as high as 68x relative to Qiskit with optimization_level=3.

IV. STAR-TO-LINE ROUTING

Often, the connectivity graph of target quantum circuits do not match the connectivity graph of target quantum hardware. Therefore, mapping and routing is required to execute the quantum circuit. Mapping refers to the initial assignment of physical qubits on the device to qubits in the quantum circuit. Routing refers to the process of inserting SWAPs that enable interaction between non-local physical qubits.

Many target quantum algorithms, such as Bernstein-Vazirani (BV), the Quantum Fourier Transform (QFT), and implementations of the variational quantum eigensolver (VQE), are dominated by instances of *star* connectivity in their program structure where a single qubit interacts with every (or nearly every) other qubit. Unfortunately, many quantum hardware platforms—especially superconducting—exhibit sparse linear connectivity between physical qubits. This motivated the development of the *star-to-line* routing pass, seen in Fig. 13 for the five-qubit BV algorithm.



Fig. 13: Star-to-line mapping used for five-qubit BV algorithm.

As depicted, the algorithm has a star connectivity, whereby q[4] interacts with every other qubit. This star connectivity can be effectively converted to a linear connectivity circuit with modest overhead. In particular, the CX between the second-to-edge qubit (q[1]) and edge qubit (q[0]) can be implemented without any routing. Then a CX and SWAP are applied between the second-to-edge qubit (q[1]) and its other neighboring qubit (q[2]). This process continues, applying a CX and SWAP between physical qubits q[i] and q[i+1], until we reach the end of the line. Moreover (not shown for brevity), the remaining sequences of CX-SWAP can be performed with just two CX gates, via standard gate cancellation identities [60], [61].

Superstaq's star-to-line router parses through a given quantum circuit to find sub-circuits which have star connectivity. Fig. 14 provides an example of such a circuit. The dashed lines indicate sub-circuits which have star connectivity.

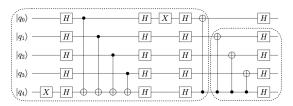


Fig. 14: Example of a circuit which contains star-connected sub-circuits.

We find the star-to-line router outperforms other state-of-theart routers. Fig. 15 plots the number of SWAP gates required to map a Bernstein-Vazirani circuit to a linear topology. The SWAP count increase quadratically with the Cirq router, but just linearly with Superstaq's router, as expected.

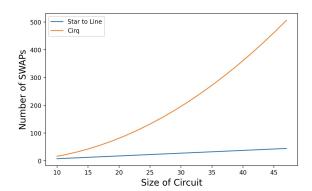


Fig. 15: Number of SWAPs vs length of circuit for Superstaq's star-to-line router and Cirq's router.

V. CONCLUSION

The premise of Superstag is that we can extract greater quantum program performance with deep cross-layer optimization tailored to the underlying hardware. In the process of building Superstaq, we advanced techniques and insights related to parametric (fractional) gates, dynamical decoupling, swap mirroring, bring-your-own gateset, Phased MicroWave decompositions, approximate synthesis, and gutrits—all of which are presented in this paper. We also find it profitable to design compilation with typical application workloads in mind, which motivates the star-to-line mapping technique. Overall, Superstag is able to achieve significant improvements in quantum program performance, exemplified for instance by our > 10x performance improvements for benchmark applications. We hope that the open-beta availability of Superstaq, through its open-source qiskit-superstaq and cirq-superstaq clients, will enable practitioners and researchers to continue to advance the frontier of what quantum computers can accomplish.

ACKNOWLEDGMENTS

We thank Woo Chang Chung, Dan Cole, David Mason, Tom Noel, and Alex Radnaev of Inflequion for their support towards executing quantum circuits on Hilbert.

REFERENCES

- [1] Leonardo DiCarlo, Jerry M Chow, Jay M Gambetta, Lev S Bishop, Blake R Johnson, DI Schuster, J Majer, Alexandre Blais, Luigi Frunzio, SM Girvin, et al. Demonstration of two-qubit algorithms with a superconducting quantum processor. *Nature*, 460(7252):240–244, 2009.
- [2] Alexander Zlokapa, Sergio Boixo, and Daniel Lidar. Boundaries of quantum supremacy via random circuit sampling. 2020.
- [3] E Urban, Todd A Johnson, T Henage, L Isenhower, DD Yavuz, TG Walker, and M Saffman. Observation of rydberg blockade between two atoms. *Nature Physics*, 5(2):110–114, 2009.
- [4] TM Graham, Y Song, J Scott, C Poole, L Phuttitarn, K Jooya, P Eichler, X Jiang, A Marra, B Grinkemeyer, et al. Multi-qubit entanglement and algorithms on a neutral-atom quantum computer. *Nature*, 604(7906):457–462, 2022.
- [5] Chris Monroe, David M Meekhof, Barry E King, Wayne M Itano, and David J Wineland. Demonstration of a fundamental quantum logic gate. *Physical review letters*, 75(25):4714, 1995.
- [6] Alexander Erhard, Hendrik Poulsen Nautrup, Michael Meth, Lukas Postler, Roman Stricker, Martin Stadler, Vlad Negnevitsky, Martin Ringbauer, Philipp Schindler, Hans J Briegel, et al. Entangling logical qubits with lattice surgery. *Nature*, 589(7841):220–224, 2021.
- [7] Joonho Lee, Dominic W Berry, Craig Gidney, William J Huggins, Jarrod R McClean, Nathan Wiebe, and Ryan Babbush. Even more efficient quantum computations of chemistry through tensor hypercontraction. PRX Quantum, 2(3):030305, 2021.
- [8] Jason Sanders and Edward Kandrot. CUDA by example: an introduction to general-purpose GPU programming. Addison-Wesley Professional, 2010.
- [9] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.
- [10] Leonardo Dagum and Ramesh Menon. Openmp: an industry standard api for shared-memory programming. *IEEE computational science and engineering*, 5(1):46–55, 1998.
- [11] Andy B Yoo, Morris A Jette, and Mark Grondona. Slurm: Simple linux utility for resource management. In Job Scheduling Strategies for Parallel Processing: 9th International Workshop, JSSPP 2003, Seattle, WA, USA, June 24, 2003. Revised Paper 9, pages 44–60. Springer, 2003.

- [12] Ryan Babbush, Jarrod R McClean, Michael Newman, Craig Gidney, Sergio Boixo, and Hartmut Neven. Focus beyond quadratic speedups for error-corrected quantum advantage. *PRX Quantum*, 2(1):010103, 2021.
- [13] Joris Kattemolle and Seenivasan Hariharan. Line-graph qubit routing: from kagome to heavy-hex and more, 2023.
- [14] Kyle E. C. Booth. Constraint programming models for depth-optimal qubit assignment and swap-based routing, 2023.
- [15] Qiskit contributors. Qiskit: An open-source framework for quantum computing, 2023.
- [16] Robert R Tucci. An introduction to cartan's kak decomposition for qc programmers. 2005.
- [17] Christopher M Dawson and Michael A Nielsen. The solovay-kitaev algorithm. 2005.
- [18] Ed Younis, Costin C Iancu, Wim Lavrijsen, Marc Davis, Ethan Smith, and USDOE. Berkeley quantum synthesis toolkit (bqskit) v1, 4 2021.
- [19] Mingkuan Xu, Zikun Li, Oded Padon, Sina Lin, Jessica Pointing, Auguste Hirth, Henry Ma, Jens Palsberg, Alex Aiken, Umut A. Acar, and Zhihao Jia. Quartz: Superoptimization of quantum circuits. In Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI 2022, page 625–640, New York, NY, USA, 2022. Association for Computing Machinery.
- [20] Easwar Magesan and Jay M. Gambetta. Effective hamiltonian models of the cross-resonance gate. Phys. Rev. A, 101:052308, May 2020.
- [21] Neereja Sundaresan, Isaac Lauer, Emily Pritchett, Easwar Magesan, Petar Jurcevic, and Jay M. Gambetta. Reducing unitary and spectator errors in cross resonance with optimized rotary echoes. *PRX Quantum*, 1(2), dec 2020.
- [22] Benchen Huang, Nan Sheng, Marco Govoni, and Giulia Galli. Quantum simulations of fermionic hamiltonians with efficient encoding and ansatz schemes. *Journal of Chemical Theory and Computation*, 19(5):1487– 1498, 2023.
- [23] Ilya G Ryabinkin, Tzu-Ching Yen, Scott N Genin, and Artur F Izmaylov. Qubit coupled cluster method: a systematic approach to quantum chemistry on a quantum computer. *Journal of chemical theory and* computation, 14(12):6317–6326, 2018.
- [24] Kaiwen Gui, Teague Tomesh, Pranav Gokhale, Yunong Shi, Frederic T Chong, Margaret Martonosi, and Martin Suchara. Term grouping and travelling salesperson for digital quantum simulation. arXiv preprint arXiv:2001.05983, 2020.
- [25] Pranav Gokhale, Teague Tomesh, Martin Suchara, and Frederic T Chong. Faster and more reliable quantum swaps via native gates. arXiv preprint arXiv:2109.13199, 2021.
- [26] Max Aksel Bowman, Pranav Gokhale, Jeffrey Larson, Ji Liu, and Martin Suchara. Hardware-conscious optimization of the quantum toffoli gate. arXiv preprint arXiv:2209.02669, 2022.
- [27] David C. McKay, Christopher J. Wood, Sarah Sheldon, Jerry M. Chow, and Jay M. Gambetta. Efficient z-gates for quantum computing. *Physical Review A*, 96(2), aug 2017.
- [28] Oinam Romesh Meitei, Bryan T. Gard, George S. Barron, David P. Pappas, Sophia E. Economou, Edwin Barnes, and Nicholas J. Mayhall. Gate-free state preparation for fast variational quantum eigensolver simulations: ctrl-vqe, 2021.
- [29] Nathan Earnest, Caroline Tornow, and Daniel J Egger. Pulse-efficient circuit transpilation for quantum applications on cross-resonance-based hardware. *Physical Review Research*, 3(4):043088, 2021.
- [30] Pranav Gokhale, Ali Javadi-Abhari, Nathan Earnest, Yunong Shi, and Frederic T Chong. Optimized quantum compilation for near-term algorithms with openpulse. In 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pages 186–200. IEEE, 2020.
- [31] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm, 2014.
- [32] David Danin and Felix Tennie. Procedure for improving cross-resonance noise resistance using pulse-level control, 2023.
- [33] Cirq Developers. Cirq, December 2022. See full list of authors on Github: https://github.com/quantumlib/Cirq/graphs/contributors.
- [34] Natalia Nottingham, Michael A Perlin, Ryan White, Hannes Bernien, Frederic T Chong, and Jonathan M Baker. Decomposing and routing quantum circuits under constraints for neutral atom architectures. 2023.
- [35] T. M. Graham, Y. Song, J. Scott, C. Poole, L. Phuttitarn, K. Jooya, P. Eichler, X. Jiang, A. Marra, B. Grinkemeyer, M. Kwon, M. Ebert, J. Cherek, M. T. Lichtman, M. Gillette, J. Gilbert, D. Bowman, T. Ballance, C. Campbell, E. D. Dahl, O. Crawford, N. S. Blunt, B. Rogers,

- T. Noel, and M. Saffman. Multi-qubit entanglement and algorithms on a neutral-atom quantum computer. *Nature*, 604(7906):457–462, April 2022
- [36] Susan M. Clark, Daniel Lobser, Melissa C. Revelle, Christopher G. Yale, David Bossert, Ashlyn D. Burch, Matthew N. Chow, Craig W. Hogle, Megan Ivory, Jessica Pehr, Bradley Salzbrenner, Daniel Stick, William Sweatt, Joshua M. Wilson, Edward Winrow, and Peter Maunz. Engineering the Quantum Scientific Computing Open User Testbed. IEEE Transactions on Quantum Engineering, 2:1–32, 2021.
- [37] Ryan Shaffer, Hang Ren, Emiliia Dyrenkova, Christopher G. Yale, Daniel S. Lobser, Ashlyn D. Burch, Matthew N. H. Chow, Melissa C. Revelle, Susan M. Clark, and Hartmut Häffner. Sample-efficient verification of continuously-parameterized quantum gates for small quantum processors. *Quantum*, accepted, 2023.
- [38] P. J. Lee, K. A. Brickman, L. Deslauriers, P. C. Haljan, L. M. Duan, and C. Monroe. Phase control of trapped ion quantum gates. *Journal* of Optics B: Quantum and Semiclassical Optics, 2005.
- [39] Andrew W. Cross, Lev S. Bishop, Sarah Sheldon, Paul D. Nation, and Jay M. Gambetta. Validating quantum computers using randomized model circuits. *Phys. Rev. A*, 100:032328, Sep 2019.
- [40] P Gokhale, ER Anschuetz, C Campbell, FT Chong, ED Dahl, P Frederick, EB Jones, B Hall, S Issa, P Goiporia, et al. Supercheq: Quantum advantage for distributed databases. arXiv preprint arXiv:2212.03850, 2022.
- [41] Yongshan Ding, Pranav Gokhale, Sophia Fuhui Lin, Richard Rines, Thomas Propson, and Frederic T Chong. Systematic crosstalk mitigation for superconducting qubits via frequency-aware compilation. In 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pages 201–214. IEEE, 2020.
- [42] Marc G. Davis, Ethan Smith, Ana Tudor, Koushik Sen, Irfan Siddiqi, and Costin Iancu. Towards optimal topology aware quantum circuit synthesis. In 2020 IEEE International Conference on Quantum Computing and Engineering (QCE), pages 223–234, 2020.
- [43] Dawei Ding, Jiachen Huang, Jianxin Chen, and Qi Ye. Compiling Arbitrary Single-Qubit Gates Via the Phase-Shifts of Microwave Pulses. In APS March Meeting Abstracts, volume 2022 of APS Meeting Abstracts, page K41.003, January 2022.
- [44] Akel Hashim, Rich Rines, Victory Omole, Ravi K. Naik, John Mark Kreikebaum, David I. Santiago, Frederic T. Chong, Irfan Siddiqi, and Pranav Gokhale. Optimized swap networks with equivalent circuit averaging for qaoa. *Phys. Rev. Res.*, 4:033028, Jul 2022.
- [45] Pranav Gokhale, Jonathan M. Baker, Casey Duckering, Natalie C. Brown, Kenneth R. Brown, and Frederic T. Chong. Asymptotic improvements to quantum circuits via qutrits. In *Proceedings of the 46th International Symposium on Computer Architecture*. ACM, June 2019.
- [46] Yunong Shi, Pranav Gokhale, Prakash Murali, Jonathan M Baker, Casey Duckering, Yongshan Ding, Natalie C Brown, Christopher Chamberland, Ali Javadi-Abhari, Andrew W Cross, et al. Resource-efficient quantum computing by breaking abstractions. *Proceedings of the IEEE*, 108(8):1353–1370, 2020.
- [47] Pranav Gokhale, Jonathan M Baker, Casey Duckering, Frederic T Chong, Natalie C Brown, and Kenneth R Brown. Extending the frontier of quantum computers with qutrits. *IEEE Micro*, 40(3):64–72, 2020.
- [48] Noah Goss, Alexis Morvan, Brian Marinelli, Bradley K. Mitchell, Long B. Nguyen, Ravi K. Naik, Larry Chen, Christian Jünger, John Mark Kreikebaum, David I. Santiago, Joel J. Wallman, and Irfan Siddiqi. High-fidelity qutrit entangling gates for superconducting circuits. *Nature Communications*, 13(1), December 2022.
- [49] A. Morvan, V. V. Ramasesh, M. S. Blok, J. M. Kreikebaum, K. O'Brien, L. Chen, B. K. Mitchell, R. K. Naik, D. I. Santiago, and I. Siddiqi. Qutrit randomized benchmarking. *Phys. Rev. Lett.*, 126:210504, May 2021.
- [50] Sergey Bravyi, Sarah Sheldon, Abhinav Kandala, David C. Mckay, and Jay M. Gambetta. Mitigating measurement errors in multiqubit experiments. *Phys. Rev. A*, 103:042605, Apr 2021.
- [51] Lorenza Viola, Emanuel Knill, and Seth Lloyd. Dynamical Decoupling of Open Quantum Systems. *Physical Review Letters*, 82(12):2417–2421, March 1999.
- [52] Nic Ezzell, Bibek Pokharel, Lina Tewala, Gregory Quiroz, and Daniel A. Lidar. Dynamical decoupling for superconducting qubits: A performance survey, July 2022.
- [53] H. Y. Carr and E. M. Purcell. Effects of Diffusion on Free Precession in Nuclear Magnetic Resonance Experiments. *Physical Review*, 94(3):630– 638, May 1954.

- [54] S. Meiboom and D. Gill. Modified Spin-Echo Method for Measuring Nuclear Relaxation Times. Review of Scientific Instruments, 29(8):688– 691, August 1958.
- [55] Palash Goiporia, Pranav Gokhale, Michael A Perlin, Yunong Shi, and Martin Suchara. Suppressing errors with dynamical decoupling using pulse control on amazon braket. aws.amazon.com/blogs/quantum-computing/ suppressing-errors-with-dynamical-decoupling-using-pulse-control-on-amazon-braket/, 2022.
- [56] Kaitlin N Smith, Gokul Subramanian Ravi, Prakash Murali, Jonathan M Baker, Nathan Earnest, Ali Javadi-Cabhari, and Frederic T Chong. Timestitch: Exploiting slack to mitigate decoherence in quantum circuits. ACM Transactions on Quantum Computing, 4(1):1–27, 2022.
- [57] Alexandre M. Souza, Gonzalo A. Álvarez, and Dieter Suter. Effects of time-reversal symmetry in dynamical decoupling. *Phys. Rev. A*, 85:032306, Mar 2012.
- [58] IBM Qiskit Development Team. Obtaining information about your backend. https://qiskit.org/documentation/tutorials/circuits_advanced/ 08_gathering_system_information.html. Accessed: 2023-05-01.
- [59] Swamit Tannu. Software Techniques to Mitigate Errors on Noisy Quantum Computers. PhD thesis, Georgia Institute of Technology, 2020.
- [60] Teague Tomesh, Pranav Gokhale, Eric R Anschuetz, and Frederic T Chong. Coreset clustering on small quantum computers. *Electronics*, 10(14):1690, 2021.
- [61] Teague Tomesh, Pranav Gokhale, Victory Omole, Gokul Subramanian Ravi, Kaitlin N Smith, Joshua Viszlai, Xin-Chuan Wu, Nikos Hardavellas, Margaret R Martonosi, and Frederic T Chong. Supermarq: A scalable quantum benchmark suite. In 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA), pages 587–603. IEEE, 2022.