

Fermilab

How to Build a Quantum Supercomputer: Scaling Challenges and Opportunities

FERMILAB-PUB-24-0843-ETD

arXiv:2411.10406

This manuscript has been authored by Fermi Research Alliance, LLC
under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy,
Office of Science, Office of High Energy Physics.

How to Build a Quantum Supercomputer: Scaling Challenges and Opportunities

Masoud Mohseni,^{1,*} Artur Scherer,² K. Grace Johnson,¹ Oded Wertheim,³ Matthew Otten,⁴ Navid Anjum Aadit,⁵ Kirk M. Bresnicker,¹ Kerem Y. Camsari,⁵ Barbara Chapman,⁶ Soumitra Chatterjee,⁶ Gebremedhin A. Dagneu,² Aniello Esposito,¹ Farah Fahim,⁷ Marco Fiorentino,¹ Abdullah Khalid,² Xiangzhou Kong,² Bohdan Kulchytskyi,² Ruoyu Li,⁸ P. Aaron Lott,^{9,10} Igor L. Markov,¹¹ Robert F. McDermott,^{4,12} Giacomo Pedretti,¹ Archit Gajjar,¹ Allyson Silva,² John Sorebo,¹¹ Panagiotis Spentzouris,⁷ Ziv Steiner,³ Boyan Torosov,² Davide Venturelli,^{9,10} Robert J. Visser,⁸ Zak Webb,² Xin Zhan,¹ Yonatan Cohen,³ Pooya Ronagh,^{2,13,14,15} Alan Ho,¹² Raymond G. Beausoleil,¹ and John M. Martinis^{12,†}

¹*Hewlett Packard Labs, CA, USA*

²*1QB Information Technologies (1QBit), BC, Canada*

³*Quantum Machines, Israel*

⁴*Department of Physics, University of Wisconsin–Madison, WI, USA*

⁵*Department of Electrical and Computer Engineering,
University of California, Santa Barbara, CA, USA*

⁶*Hewlett Packard Enterprise, TX, USA*

⁷*Fermi National Accelerator Laboratory, IL, USA*

⁸*Applied Materials, CA, USA*

⁹*USRA Research Institute for Advanced Computer Science, CA, USA*

¹⁰*Quantum Artificial Intelligence Laboratory (QuAIL),
NASA Ames Research Center, CA, USA*

¹¹*Synopsys, CA, USA*

¹²*Qolab, CA, USA*

¹³*Institute for Quantum Computing, University of Waterloo, ON, Canada*

¹⁴*Department of Physics & Astronomy, University of Waterloo, ON, Canada*

¹⁵*Perimeter Institute for Theoretical Physics, ON, Canada*

(Dated: November 18, 2024)

In the span of four decades, quantum computation has evolved from an intellectual curiosity to a potentially realizable technology. Today, small-scale demonstrations have become possible for quantum algorithmic primitives on hundreds of physical qubits and proof-of-principle error-correction procedures on a single logical qubit. Nevertheless, despite significant progress and excitement, the detailed path toward a full-stack scalable quantum computing technology is largely unknown. There are significant outstanding quantum hardware, fabrication, software architecture, and algorithmic challenges that are either unresolved or overlooked. These issues could seriously undermine the arrival of utility-scale quantum computers for the foreseeable future. Here, we provide a comprehensive review of these scaling challenges. We show how the road to scaling could be paved by adopting existing semiconductor technology to build much higher-quality qubits, employing system engineering approaches, and performing distributed quantum computation within heterogeneous high-performance computing infrastructures. These opportunities for research and development could unlock certain promising applications, in particular, efficient quantum simulation and learning/modeling quantum data generated by natural or engineered quantum systems. In order to estimate the true cost of such promises, we provide a detailed resource and sensitivity analysis for classically hard quantum chemistry calculations on surface-code error-corrected quantum computers given current, target, and desired hardware specifications based on superconducting qubits, accounting for a realistic distribution of errors. We show orders of magnitude enhancement in performance could be obtained by a combination of quantum hardware and algorithmic improvements. Furthermore, we argue that, to tackle today's industry-scale classical optimization and machine learning problems in a cost-effective manner, distributed quantum-assisted probabilistic computing with custom-designed accelerators should be considered as a complementary path toward scalability.

CONTENTS

| | |
|---|---|
| I. Introduction: Past, present, and future challenges of building quantum computers | 2 |
|---|---|

| | |
|---|---|
| A. Systems engineering for quantum hardware | 4 |
| B. Technical challenges and opportunities at different scales | 5 |
| 1. Challenges at 100–1000 physical qubits | 5 |
| 2. Challenges at 1000–10k physical qubits | 6 |
| 3. Challenges at 10k–100k physical qubits | 7 |
| 4. Challenges at 100k–1M physical qubits | |

* masoud.mohseni@hpe.com

† john@qolab.ai

| | | | |
|---|----|--|----|
| and beyond | 7 | VIII. Discussions | 48 |
| C. A full-stack hardware–software architecture for high-performance quantum computation | 9 | A. Supply chain management toward utility-scale | 48 |
| II. Toward high-quality quantum hardware and high-performance control | 10 | B. Exponential scaling progress: A mirage or reality? | 49 |
| A. Qubit fabrication | 11 | References | 50 |
| B. Wafer-scale integration | 13 | A. Analysis of logical circuits for quantum resource estimation | 55 |
| C. Control hardware | 15 | 1. Workflows for generating logical circuits | 55 |
| III. Toward fault-tolerant quantum computation | 18 | 2. Analysis of Trotter errors and their propagation into phase estimation | 56 |
| A. Fault-tolerant circuit compilation | 20 | 3. Propagation of errors in qubitization | 59 |
| B. Benchmarking quantum hardware for the quantum memory experiment | 21 | B. Quantum resource estimations using TopQAD | 60 |
| C. Sensitivity of FTQC performance to specific hardware improvements | 22 | 1. The compilation process | 60 |
| D. Impact of qubit and gate quality distributions on logical error rates | 23 | 2. The assembly process | 60 |
| E. Emulation of other FTQC protocols | 24 | 3. Comparison with the AzureQRE toolkit | 62 |
| F. High-performance real-time decoding platform | 27 | C. Double-factorized quantum chemistry | 62 |
| G. Distributed FTQC across multiple dilution refrigerators | 29 | D. Runtime of classical algorithms for quantum chemistry | 62 |
| IV. Resource estimation for fault-tolerant quantum computation | 30 | E. Circuit-level noise model | 63 |
| A. Quantum computation of electronic spectra as a representative high-utility application | 30 | I. INTRODUCTION: PAST, PRESENT, AND FUTURE CHALLENGES OF BUILDING QUANTUM COMPUTERS | |
| B. Quantum resource estimation studies for <i>p</i> -benzyne and FeMoco | 32 | | |
| V. Toward high-performance hybrid quantum–classical computing | 35 | Quantum computing has seen remarkable progress over the past few decades despite facing tremendous conceptual, theoretical, and technical challenges. There have been a few significant breakthroughs, such as Shor’s algorithm for integer factoring [1] to solve a seemingly exponentially hard classical problem, or the invention of quantum error-correction (QEC) [2, 3] to tackle the fundamental problem of <i>decoherence</i> , without which Shor’s algorithm would remain merely a mathematical curiosity. Starting from small experiments that manipulate single- or few-qubit systems, research groups using a variety of technologies can now make and operate quantum processors with on the order of 100 physical qubits. Some proof-of-principle speedups over conventional (classical) supercomputers called “quantum supremacy” [4] or “quantum advantage” [5] have been demonstrated, but only for carefully crafted problems. The next step is to scale up quantum processors to demonstrate significant speedup for a practical problem in a cost effective manner, thereby achieving “quantum utility” [6]. | |
| A. HPC programming environment extensions | 37 | Recently, there has been a significant interest in the high-performance computing (HPC) community in employing quantum computation as a complementary paradigm beyond exascale supercomputing [7–9]. Rather than replacing classical computers as general-purpose processors, quantum computers can be better understood | |
| B. High-performance quantum workload distribution | 39 | | |
| C. High-performance quantum–classical workload scheduling | 41 | | |
| VI. A near-term application: Distributed quantum simulation | 42 | | |
| A. Multi-GPU: Dynamical quantum phase transitions of 2D transverse-field Ising models | 43 | | |
| B. Multi-QPU: Strongly disordered quantum spin glasses | 44 | | |
| VII. Toward heterogeneous quantum and probabilistic computing | 46 | | |
| A. Probabilistic computing with intrinsic higher-order interactions | 46 | | |
| B. Hardware implementation of p-computers | 47 | | |
| C. Scaling up p-computers: a distributed approach | 47 | | |
| D. Quantum-assisted probabilistic computing with custom-design accelerators | 48 | | |

as accelerators or coprocessors that can efficiently carry out specialized tasks within an HPC framework. Hybrid quantum-classical frameworks will be crucial not only in the near term—the noisy, intermediate scale quantum (NISQ) era [10]—but also for future fault-tolerant quantum computation (FTQC), as error-correction schemes will rely heavily on classical HPC and the number of logical qubits will be fairly small for the foreseeable future. To achieve true utility-scale quantum computing, successful integration with existing heterogeneous HPC infrastructures and the development of a hybrid quantum-classical full computing stack are necessary.

There are a number of challenges to successful quantum-HPC integration. At the hardware architecture and system design levels, there are significant differences between the quantum and classical components in physical scale, hardware reliability, control electronics, communication bandwidth, and time scales of operation. At the algorithmic level, the challenges lie within memory access, data sharing and movement, and information extraction. To build a quantum-centric supercomputer [7, 8] at scale, much better quantum components must be built at different layers that ultimately rely on much higher quality qubits. While basic research remains critical, a comprehensive engineering approach targeting the full stack must be taken in parallel, with the aim of steadily increasing the technology readiness level (TRL) of the components at various levels of the full stack. This mindset is needed for NISQ devices, logical intermediate-scale quantum (LISQ) processors, and for FTQC.

Utility-scale quantum computing ultimately involves very deep quantum circuits requiring logical error rates far below those experimentally feasible with physical qubits and gate operations. Quantum error correction [2, 3] is necessary, as even very small errors rapidly accumulate, resulting in significant errors. Similar to classical error correction, QEC utilizes redundancy in the encoding of information to detect and correct errors. However, unlike in the case of classical telecommunication, copying quantum information is prohibited due to the no-cloning theorem [11]. In addition, quantum measurements irreversibly collapse the wavefunction. Therefore, the key mechanism for error correction without destroying logical information is through exploiting quantum entanglement with ancillary subsystems. Quantum error-correction codes (QECC) protect a smaller logical state of computation within a much larger highly entangled quantum state involving many noisy physical qubits. Therefore, errors below a certain weight can be detected and corrected. Multi-qubit stabilizer measurements produce a set of syndromes that detect whether some of the physical qubits have been corrupted. A QEC decoder can then infer the most likely errors for the detected syndromes. With this knowledge, corrective operations can be either physically applied (active error correction) or kept track of in software.

While QEC is necessary, it is by no means sufficient for reliable large-scale quantum computation. When per-

forming QEC, the operations on the physical qubits necessary to implement the QECC can eventually themselves add more errors than they are able to correct. Thus, achieving FTQC requires implementation schemes in which QEC succeeds in suppressing errors faster than it causes them. According to the threshold theorem, once the physical errors of the quantum system are below a certain threshold, the overhead of these schemes scales poly-logarithmically with the precision of computation [12–14]. However, these schemes introduce a substantial overhead in physical resources, the estimating and validating of which is a critical step toward practical realization of FTQC at utility scale and the cost of doing so.

Here, at the lowest level of the full stack—the physical components for storing and processing quantum information—we focus on superconducting qubits. Historically, the success of superconducting qubits has come from a sustained focus on improving qubit coherence using different approaches. At present, although there are a variety of ideas on how to make further advances, there is sufficient justification for using advanced semiconductor processing and tools. This approach will need to be guided by and integrated with more complex device architectures and structures, such as bump-bonding and advanced packaging, which are designed around specific decoherence mechanisms such as large two-level state loss from amorphous insulators.

These advances were also shaped by an understanding that qubit architecture would be improved by designs that turned off the qubit-qubit interaction, which does not occur naturally using simple coupling capacitors. The adjustable coupler was pioneered by UCSB and Google and is an example of careful systems engineering that was initially assessed skeptically by a majority of the superconducting qubit community. Although the adjustable coupler was significantly more complex, as it required using a qubit to turn interactions on and off, the reduction in crosstalk enabled Google to achieve the quantum supremacy milestone [4]. Today, the adjustable coupler has been adopted and integrated into superconducting systems from IBM, USTC, Rigetti, and IQM. Alternative superconducting qubit architectures such as the fluxonium are only being realized [15] thanks to the adjustable couplers technology. Further innovations which, like adjustable couplers, trade simplicity for performance are needed to push errors in large qubit systems to the 10^{-4} range.

Despite the large size of superconducting qubits, roughly millimeters, the numbers of qubits have been scaled up in a brute-force manner once single- and few-qubit systems were established. Indeed, recent advances enabled building large enough quantum computers to test system performance and demonstrate the execution of algorithms. Experiments have even achieved logical error rates at the 10^{-10} level, although surpassing this rate appears to be obstructed by errors induced via cosmic rays [16], which demands further investigation. Although it

has been speculated that the large footprint of superconducting qubits makes them too difficult to scale, in this position paper we describe how modern semiconductor processing may be used to solve this problem.

While we introduce a definite architecture for coupling transmon qubits, alternative designs might also improve performance. Our expectation is that advanced fabrication targeted to improve both coherence and scaling could be used for a variety of approaches. We believe these ideas will greatly enhance the likelihood of making useful quantum computers.

A. Systems engineering for quantum hardware

To make these advances, the most important concept is systems engineering, where one embraces the idea that many system parameters must be simultaneously optimized for a complex system. A significant obstacle in prior research stems from treating quantum computation as a highly tailored quantum physics experiment, with physicists naturally focusing on isolating physical phenomena in order to fully understand each of them. For example, tables of metrics often emphasize the best quality of the various approaches, whereas system engineering typically constrains the system with its worst quality.

There are many systems engineering parameters that must be considered and they differ for each technology. Here, we focus on the four most important parameters that serve as a concise but powerful way to compare quantum hardware approaches.

Quality. Qubits are different from classical bits in that they are fundamentally prone to errors. This is in part due to their analog-control nature, but also due to their quantum properties, such as decoherence (i.e., entanglement of the qubits with their environment). It is important to note that the scalability of an approach, expressed by the number of qubits and the length of circuits successfully executed, is now mostly bottlenecked by qubit errors. For example, a 50-qubit system with 1% errors will allow only about two layers of gates (at 50 qubits per layer) before there is an error in the execution of the circuit: this clearly limits the system’s utility. Errors in the range 0.01–0.1% are believed necessary for both NISQ and FTQC algorithms, as is described in more detail in Section II A.

Quantity. This is the most intuitively understood metric. As discussed above, at first we need to optimize for low error rates. However, as errors reach the 0.01–0.1% range, the ability to scale up the number of qubits becomes more important. This is, in part, because qubit number scales logarithmically with the error suppression rate of QECCs such as the surface code. So, once the error rate is sufficiently low, the strategy should change to prioritizing qubit counts. Scaling to thousands or millions of qubits is required in the long term, and thus careful deliberation is needed on how to achieve that with

any particular technology.

Speed. The clock speed of qubits is often not being highlighted in public roadmaps, which put emphasis instead on harnessing quantum advantage from “exponential quantum parallelism”. Although this Hilbert space size advantage can be argued for theoretically, we show in our resource estimations (Section IV) that speed is crucially important for practical utility. This is because the speed of various qubit technologies differs greatly, by up to four orders of magnitude. For example, superconducting qubits have typical single- and two-qubit gate times of 50 ns or less, whereas typical clock speeds of atomic systems are ~ 100 μ s, often limited by the time scale of mechanical motions of atoms or ions. In addition, superconducting gates are operated in parallel, whereas ion-trap systems have their primitive gates often processed serially through interaction zones in leading quantum-charge-coupled-device (QCCD) architectures. For NISQ applications with short circuits, repeated trials are typically necessary to obtain sufficient statistics, and thus end-to-end experiments are very slow even for superconducting qubits. Generally, one can understand the need for speed by noting that a $1000\times$ increase in speed translates to $1000\times$ more throughput. However, for FTQC, a $1000\times$ slowdown can render certain applications impractical because, even with the fast clock-speed of superconducting qubits, the execution time of utility-scale algorithms can be on the order of months or years (see Section IV for predicted execution times).

Connectivity. The number of connections from one physical qubit to another is also an important metric, and has an interesting trade-off with speed. For neutral atom and trapped-ion systems, the connectivity is generally considered all-to-all, but this may be the case only for small enough systems, e.g., within a single trap. This all-to-all connectivity also typically requires ample time for shuttling qubits. Superconducting qubits do not need to move but have sparse connectivity; they typically have nearest-neighbor interactions, either to 4 qubits in a regular rectangular lattice or 2.5 qubits on average for the heavy-hex lattice [17]. This more-limited connectivity can be factored into the design of near-term algorithms, and thus it is hard to make a fair performance comparison with respect to connectivity. For a fault-tolerant quantum computer, the present connectivity of superconducting qubits is sufficient to support error-corrected logical qubits such as the surface codes. Additionally, the more-connected rectangular lattice architecture is perceived to be less degraded by qubit dropouts, an important system constraint.

Tying the four parameters together. A quantum computer must incorporate a large number of qubits that are well-connected by sufficiently fast high-quality gates. Performing well with respect to all the above metrics is necessary for creating highly entangled quantum states between the qubits. However, there may be trade-offs between these metrics; e.g., with lower connectivity, more gates are needed to entangle qubits, and if these gates

are too slow, decoherence may limit the amount of entanglement generated. Given that entanglement is necessary for quantum computational advantage, *the size of the entangled states that can be prepared* by the computer is a useful system-level metric that incorporates the four parameters above and their trade-offs [18].

Next, we discuss various quantum hardware, software, and algorithmic challenges that one would face when scaling the system size from 100 physical qubits for NISQ processors to beyond 1 million qubits required for utility-scale applications on fault-tolerant quantum computers. These challenges, in turn, offer significant research and development opportunities.

B. Technical challenges and opportunities at different scales

The success of quantum computing at large scale will require overcoming major obstacles. Notably, much of the current practical know-how is for NISQ computers, complemented only by theoretical developments for larger scales. As quantum processors increase in size, with physical qubits from ~ 100 for NISQ to $\sim 10^7$ for utility-scale FTQC (corresponding to $\sim 10^4$ logical qubits, given QEC overheads), challenges of very different natures emerge at each scale. Such challenges can be mitigated with ad hoc approaches at small scales [19], but require fundamentally new solutions for true scalability. To benchmark a quantum computer consisting of 1–10 million physical qubits, innovations are required at intermediate scales. Thus, it is necessary to develop a comprehensive multi-scale roadmap for superconducting qubits that will tackle these challenges and provide corresponding testing, validation, and benchmarking at each scale. This roadmap must address quantum device design, fabrication, control electronics, calibrations, and interconnects. Various classes of noise sources such as T_1 , T_2 , single- and two-qubits errors, crosstalk, $1/f$ noise, two-level systems (TLS) defects, fat-tail of error distributions, background radiation, and low-fidelity interconnects must be characterized and dealt with at their relevant scales.

Ultimately, efforts to scale the number of physical and logical qubits in quantum processors must rely on QECCs. We foresee that different scales, and possibly different applications, may favor different types and sizes of error-correcting codes. Even in the case of surface codes, different variants (e.g., the XZZX or XY codes [20, 21]) may be preferable for different hardware noise profiles. These codes must be supported by architectural provisions for fast classical decoding and control based on syndrome measurement results. Additional support is required at the operating and compilation level.

Although many of the scaling challenges are rooted in device and architecture research, we acknowledge the vacuum for impactful applications of quantum computing in its current state. Utilizing quantum computers to

perform useful tasks such as quantum simulation reveals an additional set of scaling challenges, including problem identification and data management, e.g., loading, pre- and post-processing, and scheduling.

Here, we highlight some important challenges at four different scales characterized by the number of physical qubits. This multi-scale approach not only categorizes known challenges, but reveals untold or overlooked challenges that could present significant stumbling blocks to building useful and cost-effective quantum computers. For each scale we describe the challenges in a bottom-up order, from qubit fabrication, to hardware control, calibration, error correction, hybrid quantum–classical co-processing, micro- and instruction-set architectures, and finally algorithms and applications.

1. Challenges at 100–1000 physical qubits

At the intermediate scale, key challenges involve individual qubits and gates operating within the system, as well as fabrication and basic operation [22]. At this scale, executing quantum algorithms is used primarily to demonstrate and characterize hardware capabilities.

Fat-tail distribution of errors. The subtlety of decoherence mechanisms for superconducting qubits is underappreciated because researchers often report the coherence times of their best qubits. Measuring the median is clearly better, but reporting the worst 1% would be a more faithful reflection of system performance at scale. Indeed, for published Google and IBM data, the worst 10% of the T_1 data drops significantly ($30\text{--}100\times$) away from a Gaussian distribution (see Section III D for further analysis of these effects).

Qubit fabrication. We have recently experimentally determined (Section II A) that better fabrication can improve T_1 tails so that a smaller fraction of qubits show degradation, and the drop in T_1 is smaller. This data points to the fact that better benchmarking and process control is needed for superconducting fabrication. This is especially important for cryogenic quantum devices, as low-temperature testing is much more difficult than wafer probing of semiconductor devices at room temperature. We discuss process control and component-level testing in Section II A for qubit fabrication.

Recalibration. Another overlooked technological risk is that coherent TLS defects fluctuate in time, requiring recalibration of the quantum computer. Today, with systems consisting of 100 qubits, full recalibration is needed approximately once per day and can take up to two hours, even though leading methods for QPU calibration involve representation as a directed acyclic graph [23], which is amenable to GPU-accelerated and reinforcement learning-based approaches [15]. Because the rate of emergence of outlier qubits with low coherence is proportional to the number of qubits, a 1000-qubit computer becomes effectively unusable because it requires constant recalibration. We discuss how to reduce the TLS defects

to improve coherence, two-qubit error rates, and outlier emergence in Section II A.

Catastrophic error bursts. A technical challenge recently revealed by a Google experiment on error correction is the impact of cosmic rays on qubit error rates [16, 24]. Although this may impose a lower bound on the error rate of logical superconducting qubits at the $\sim 10^{-10}$ range with the help of gap engineering, additional mitigation strategies [25] are described in Section II A.

Real-time decoding. At this scale, it should be possible to create logical qubits and benchmark real-time error correction. The challenge of performing real-time error correction for superconducting qubits is the speed at which the qubits operate. Today, state-of-the-art decoders for superconducting qubits take $\sim 60 \mu\text{s}$ to decode $d = 7$ surface codes [26]. Smaller fast-feedback experiments have demonstrated a decoding response times of $9.6 \mu\text{s}$ [27]. However, at $\sim 0.5\text{-}\mu\text{s}$ -long stabilization rounds, the total latency inclusive of decoding and redirecting the waveform in an FPGA needs to be within $\sim 5\text{--}20 \mu\text{s}$ for code distances obtained in our resource estimation studies (see Table IV) to avoid compilation bottlenecks [28]. Even faster decoding is desirable to eliminate more sources of coherent and incoherent errors. See Section III E for further discussion on the effects of decoder delay and Section III F for details on an approach to fast and tightly integrated real-time decoding with petaflop/s processing speed.

Circuit knitting overhead. In the past few years, circuit knitting methods have been introduced to allow running quantum circuits that require more qubits than are available on a single processor at the current scale [29–31]. Formally, these methods incur an exponential classical post processing overhead for exact reconstruction of a quantum observable. To enable distributed quantum circuit execution at scale, innovative techniques must be devised for reducing this exponential overhead. While the challenge of quantum workload distribution emerges at the scale of 100–1000 physical qubits, it will be present at all later scales. In Section V B we provide a detailed discussion on this topic as well as present a family of adaptive circuit knitting methods. In Section VI B, we show a particular implementation of this adaptive circuit knitting approach that could significantly reduce overheads for quantum simulation of quantum spin glasses via an approximate tensor-network contraction over distributed quantum circuits.

NISQ computing. Systems consisting of 100–1000 physical qubits create unique challenges for NISQ algorithms. First, the larger number of qubits requires a higher shot count. This is due to the fact that many variational algorithms produce information spread across multiple qubits and the output quantum states are not localized to a small number of qubits. The second challenge is that for potentially useful applications, e.g., simulating quantum dynamics, typically one needs more than 100+ qubits at depths larger than what can be achieved by a 10^{-3} two-qubit error rate. We discuss how to ad-

dress these challenges in Section V on high-performance quantum–classical coprocessors.

More recent developments in NISQ algorithms have utilized the notion of adaptive circuits, where mid-circuit measurements and feed-forward information are used to reduce circuit depth [32–34]. Making use of such constructions will require the ability to make rapid measurements and, within the coherence time of the qubit, perform additional operations based on the measured results. For certain applications of quantum computing such as calculating the ground-state energy of a chemical system, extensive preprocessing is necessary to formulate the problem in a way amenable to quantum computers, even for small but challenging systems. For example, identifying the proper active space for the iron-molybdenum cofactor (FeMoco), which has long been hailed as a premier application of quantum computing [35], is itself a complicated computational task [36]. Integrating the quantum computer in an HPC environment can help mitigate these issues (see Section V).

We also note that, in the near-term with qubit counts below 1000, automated testing techniques at the system and component level are necessary. We discuss these procedures in Section II A on qubit fabrication for component-level testing.

2. Challenges at 1000–10k physical qubits

At the large scale, system integration and orchestration challenges become more prominent, including those related to high power consumption and costs, and availability of established fabrication technologies [22]. At this scale, algorithmic benchmarking becomes necessary to assess and optimize performance.

Wiring and packaging. Beyond 1000 qubits, a new under-appreciated systems challenge emerges, that of how to compactly address wiring, control, and circulation within today’s dilution refrigerators. A secondary aspect is the opportunity to drastically reduce the cost. For example, a cryostat for a 150-qubit processor with coaxial wires is \$5M, with \$4M devoted to wiring alone. Without circulators, $10\text{--}100\times$ more qubits can fit into a single dilution refrigerator. This will allow the packing of 20k qubits on a single $14\times 14\text{-cm}$ die. However, with new packaging of 1000–10k wires, crosstalk will likely be a dominant hardware error, requiring new designs based on electromagnetic simulations. Regrettably, state-of-the-art electromagnetic simulations have been validated only on the order of six qubits. We discuss these issues and mitigation opportunities in Section II B (wafer-scale integration), including scaling up crosstalk simulations to thousands of qubits.

Control electronics. The ability to control several thousands of qubits is necessary, but it would drive up both the cost of the electronics and the total thermal budget required for the control electronics, which in turn would increase cooling costs. We discuss opportunities

to reduce both costs and power consumption for classical CMOS control in Section II C. We also discuss the need for advanced qubit calibration, which will be necessary even with improved qubit fabrication.

The largest risk of this phase is the cost of development of these processes, which could be mitigated by leveraging the semiconductor industry. In Section II B on wafer-scale integration, we also discuss how to leverage the existing semiconductor industry to drastically reduce costs (see also Section VIII A). Because of the high costs of developing, building, and operating fault-tolerant quantum computers, there must be a strong emphasis on understanding the impact of exact hardware noise profiles on the choice of error correcting codes, as well as the resulting resource estimates for useful applications with utility-scale value. In Section III B we explain how we use hardware noise profiles at this scale to inform FTQC compilation and assembly at the utility scale.

Near-term applications. Algorithmically, the problem of data input and output starts to become challenging at the scale of 1000–10k physical qubits. Target problems at this scale could require a large amount of classical data to either be loaded onto the quantum computer or to be read from the quantum computer. Both the classical processing of this data and the quantum resources (circuit depth or measurements) can grow quickly. Without quantum error correction, quantum computers at this scale will not be able to execute standard fault-tolerant quantum algorithms such as quantum phase estimation or Shor’s algorithm. However, they will be capable of executing relatively deep circuits that are well beyond anything classically simulable, even with approximations. Therefore, there is an opportunity for discovering heuristic quantum algorithms that could provide potential utility. Rigorously benchmarking such algorithms against the classical and HPC-accelerated state of the art will be necessary to convincingly demonstrate their accuracy and effectiveness. As such, good, hardware-agnostic benchmarks in various application domains (like chemistry, materials science, and optimization) are necessary to enable testing newly discovered heuristic quantum algorithms.

3. Challenges at 10k–100k physical qubits

At the very large scale, circuit-level scaling challenges become significant [22, Table 1], including verification, testing, and debugging. For conventional integrated circuits, the challenge of “dark silicon” arises, where a significant fraction of the chip performs various service roles. In quantum computing, FTQC creates a similar overhead.

FTQC overhead. A major challenge at this scale is reducing the cross-talk noise and two-qubit gate errors. Unfavorable scaling of accumulated errors can increase the overhead of QECCs needed to compensate for them, further undermining quantum advantage. This issue is

the focus of our device-level efforts to mitigate errors (see Section II B on scaling cross-talk simulation), but it can also be addressed at the architecture level.

At the scale of tens of thousands of physical qubits, many fault-tolerant protocols including full-fledged magic state distillation units can be implemented and validated. Yet, the high space and time overhead of FTQC mean this scale will still fall short of demonstrating quantum utility. This prompts the need for advancements in QEC and FTQC schemes that reduce the overhead of fault tolerance, a goal actively pursued in current research trends for building “good” QECCs, i.e., those with high encoding rates, such as the quantum LDPC codes [37].

Moreover, the successful realization of QEC requires low-latency integration of QPUs with GPUs, which are very effective in executing a large number of identical, relatively shallow computations in parallel on different input data. Such advantages are relevant to real-time decoding, as discussed in Section III F. Another approach to relaxing the requirements of decoders is construction of better codes with faster decoding algorithms. It has been speculated that there may exist QECCs whose decoding time is independent of the code distance [38].

Verification, testing, and debugging. As QEC circuits become more sophisticated and undergo optimization to reduce overhead, the possibilities for introducing design errors during these optimizations increase. Verification aims to catch design bugs as soon as they are introduced, testing looks for problematic behaviors in a physical quantum computer (by running specific circuits), and debugging attempts to diagnose and correct problems. Historically, each of these steps became a bottleneck to scaling of classical semiconductor circuits, and required the development of new algorithmic technologies and hardware solutions to sustain scaling [39].

These tasks are much more complicated for quantum circuits than classical ones. For example, a quantum counterpart to the conventional equivalence-checking technique [40] must tackle unitary operators acting on exponentially large Hilbert spaces. Similarly, testing must be heavily optimized to handle the large number of trials required in view of the non-deterministic nature of quantum measurements and the frequent need of QPUs for recalibration [41]. More-sophisticated approximate testing [42] techniques must also be developed to take the error tolerance of quantum computation into account. Finally, it is much more complicated to diagnose and eliminate errors [43]; therefore, more-scalable debugging techniques are needed which can benefit from HPC hardware support.

4. Challenges at 100k–1M physical qubits and beyond

Computation at extreme scales faces system-level and complexity-theoretical challenges as well as those related to physical embedding and distributed computation as

quantum interconnects become a bottleneck [22, Table 1]. For quantum computers dominated by QEC, these challenges take on specific forms, as described below. Additionally, finding “killer apps” for quantum computers and validating their performance remains challenging.

Distributed FTQC. To achieve utility scale, tens to thousands of logical qubits are needed (Table III). Even at a 10^{-4} two-qubit error rate, this translates to 1 million (or more) physical qubits, which is about an order of magnitude more than what can practically be placed in today’s dilution refrigerators (DR). Significantly larger DRs will be costly. Therefore, performing large-scale FTQC will likely require quantum interconnects between multiple DRs. Optical interconnects have been proposed for providing such quantum links between distinct DRs [44–48]. We discuss the first analysis of the effects of noisy optical interconnects for QECCs on superconducting computers in Section III G and discuss the compilation of FTQC algorithms on a multi-DR architecture in Section III G.

Given the length of utility-scale FTQC algorithms and the need for frequent recalibration of QPUs, a quantum operating system (QOS) must manage an excess supply of on-boarded and off-boarded DRs mid-runtime and dispatch quantum characterization, verification, and validation (QCVV) protocols on them with an appropriate cadence. This reveals a fundamental difference between FTQC compilers and the conventional compilers for classical computers: classical compilers do not require knowledge of the noise profile of the hardware, whereas the choice of QEC codes, their sizes, and ancillary modules (e.g., magic state factories) all require detailed information about the hardware noise characteristics. We describe some preliminary steps toward addressing these challenges for FTQC compilers in Section III A.

Although it has been hypothesized that one can use optical interconnects between DRs to entangle qubits, the technology is still in its infancy. Therefore, in Section VI we show how adaptive circuit knitting strategies could delay the need for optical interconnects via high-performance distributed quantum evaluation of subcircuits and classical post-processing to merge the solutions. With large-scale integration of a quantum accelerator with a classical supercomputer, dynamically dispatching workloads to and from the quantum computer will become a complex challenge. We will discuss the use of near real-time circuit synthesis and dispatching in Section V C on high-performance quantum–classical workload scheduling.

Micro-architecture standardization. Efficient compilation and execution of large FTQC programs demands optimized and modular instruction-set architectures that are drastically different from that of conventional computers. A challenge at this scale is converging to optimal and standard micro-architectures for the computer. Recent studies [28, 49–51] suggest that efficient instruction pipelines for FTQC using solid-state qubits comprise (potentially multi-level) magic state factories

for producing high-quality resource states for consumption in a memory unit (which we call the core processor; see Section III A). Therefore, unlike the conventional von Neumann architecture wherein data is taken from memory blocks to the gates, in the FTQC pipelines, high-quality gates are brought to the memory block. Thus, FTQC micro-architectures may better resemble that of in-memory computing technologies [52]. This suggests a rethinking of conventional memory hierarchies (e.g., cache, L1, and L2) for quantum memory blocks. Notably, quantum random access memory (QRAM) [53] is used in quantum computing literature as means for accessing and manipulating classical or quantum data in superposition. However, monumental resources are required to insure its fault tolerance [54]. Quantum LDPC codes may provide a path forward for realization of feasible quantum memory blocks.

Quantum algorithm discovery. The absence of efficient quantum memory blocks and methods for reading from and writing into them is one of the reasons for the lack of useful quantum algorithms for conventional enterprise problems involving classical big data, for which classical AI has provided major breakthroughs at an increasing pace. This read–write bottleneck eliminates the exponential speedups promised by many algorithms, e.g., quantum machine learning (QML) applied to classical data. Such algorithms include quantum linear system solvers [55, 56], quantum clustering [57], quantum principal component analysis [58], and quantum support vector machines [59]. However, the promise for exponential quantum advantage still holds for quantum data [60] and substantial progress has been made towards overcoming the known trainability limits of quantum neural networks [61].

Moreover, processing classical data, e.g., via coherent arithmetic operations, is very costly for quantum computers [62, 63]. Indeed, the qubitized electronic-structure quantum simulation for which we provide resource estimates in Section IV uses look-up tables (coined as quantum read-only memory, or QROM) to avoid calculating trigonometric functions [64]. In general, applications with classical inputs and outputs are fundamentally limited because quantum computers do not provide a universal advantage. For example, there is provably no quantum advantage for standard comparison-based sorting — no quantum algorithm can solve the task asymptotically faster than conventional algorithms do [65].

Even if a given subroutine in a important application is accelerated by a quantum oracle, it must be a single distinctive bottleneck, otherwise the impact of quantum speedup will be capped by Amdahl’s law [66]. Shor’s algorithm for integer factorization and its variants offer a rare combination of likely exponential quantum speedup with a practical need for running the algorithm on many different inputs. In general, quantum optimization could offer a quadratic speedup [67] with many implicit assumptions such as lack of explicit structures in the problem instances. However, the opportu-

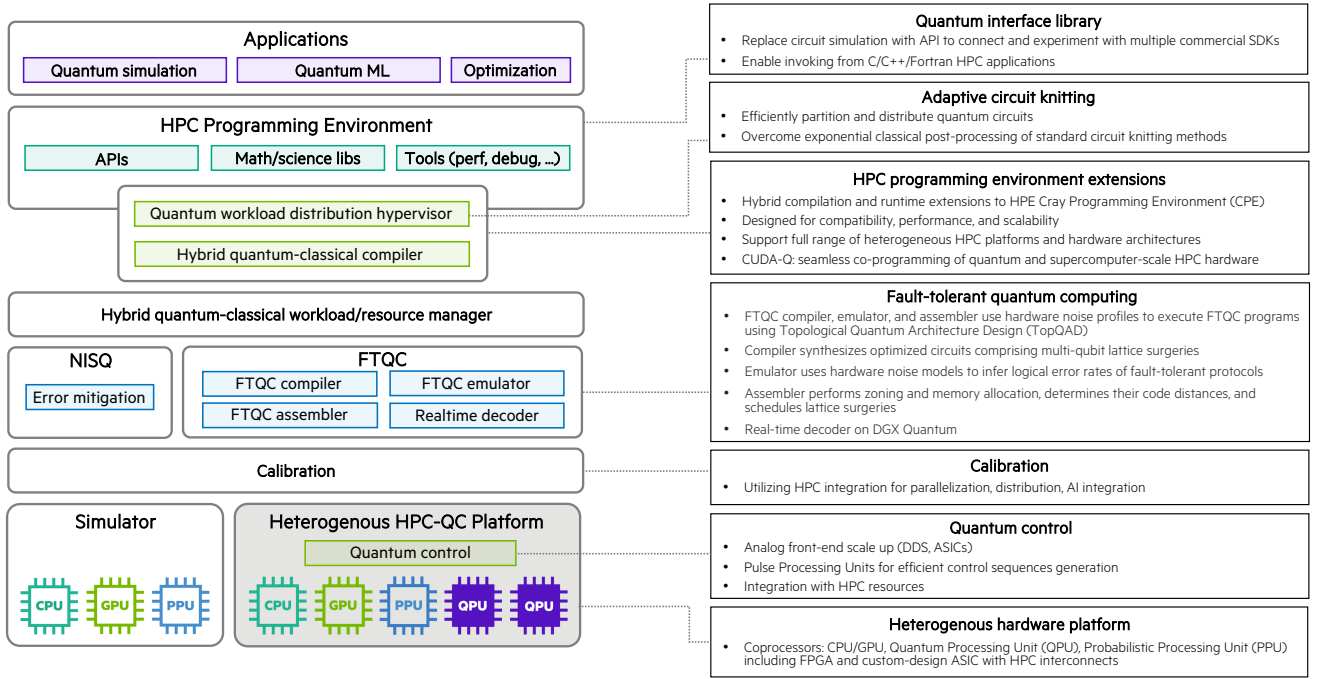


Figure 1: Architecture diagram of a quantum-classical full-stack solution. Extensions within the HPC programming environment include a quantum interface library for seamless invocation of quantum kernels, an adaptive circuit knitting hypervisor for efficient quantum workload partitioning and distribution, and quantum compiler and runtime extension for performant quantum circuit compilation. A customized hybrid workload manager ensures maximal quantum resource utilization in a multi-user environment. For fault-tolerant quantum computation, a compiler, emulator, assembler, and real-time decoder work together to use hardware noise profiles to synthesize optimized fault-tolerant circuits. Calibration and control of quantum resources use specialized hardware and are integrated with HPC. At the hardware layer, heterogeneous coprocessors include CPUs/GPUs, quantum processing units (QPUs), probabilistic processing units (PPUs), FPGA and custom-design ASIC with high speed low-latency scale-up interconnect.

nity for such quadratic speedups in practice might be slim and most likely eventually washed away at scale by the huge overhead of QEC [68]. Engineering at scale requires developing novel quantum heuristic algorithms that work in concert with their classical counterparts, see Section VII for a discussion on the possibility of accelerating classical probabilistic sampling by quantum fluctuations. Recently, there has been tremendous interest on the family of quantum approximate optimization algorithms (QAOA) [69, 70], including numerical or theoretical studies that claim potential scaling advantages might be possible [71–74]. However, one has to be careful with small-sized effects or the contrived nature of many benchmarking problems that could hinder a true scaling advantage in practical scenarios for which highly tuned classical heuristics are available. Historically, it has proven difficult to develop quantum heuristics that could stay relevant at scale. The search for new quantum algorithms on classical inputs remains a major avenue of research.

Validation of quantum algorithms. Even for quantum mechanical problems for which quantum computation is envisaged to provide revolutionary quantum advantage [58, 60], the validation of such algorithms—ensuring that they produce correct outputs, especially in practice—remains a challenge. For example, quantum computation of electronic spectra of molecules relies on preparation of input states with significant overlap with

the ground state (whose energy is to be estimated). However, is it unclear whether the commonly adopted choices (e.g., the Hartree–Fock state) will be sufficient. Indeed, ground state preparation is a QMA-hard problem, and even the FTQC quantum algorithms for such tasks are merely heuristics that may fail in practice. Motivated by these challenges recent studies have focused on efficient preparation of better initial states for such tasks [75].

Supply chain. The final challenge to address is the cost of a quantum computer, which could be reduced by leveraging the existing semiconductor supply chain. In the final Section VIII A, we will discuss how some of the leading fabrication, chip manufacturing, and system integrator companies, e.g., Applied Materials, Synopsys, Nvidia, and HPE, could establish a supply chain to drive down costs.

C. A full-stack hardware–software architecture for high-performance quantum computation

In order to tackle some of the key technical challenges listed above, here we introduce a heterogeneous quantum-classical full-stack hardware and software system architecture [9]. We outline how one could adopt existing semiconductor ecosystems and conventional high-performance infrastructures to build such an architec-

ture, which is schematically illustrated in Figure 1 (see also Section V). At the highest layer, an HPC programming environment is extended to include a quantum accelerator API consisting of an interface library for seamless invocation of quantum kernels, an adaptive circuit knitting hypervisor for efficient quantum workload partitioning and distribution, and a hybrid quantum–classical compiler. A hybrid quantum–classical workload manager ensures optimal quantum resource utilization in a multi-user environment. To support fault-tolerant quantum computation, a compiler, emulator, assembler, and a real-time decoder together use known hardware noise profiles to synthesize optimized fault-tolerant circuits to solve the problem at hand. Calibration and control of quantum resources use specialized hardware that are integrated with the HPC. Heterogeneous coprocessors—including CPU/GPUs, quantum processing units (QPU), probabilistic processing units (PPU)—allow the system to partition a particular problem into subproblems that can be sent to the appropriate coprocessors.

Current approaches to building a quantum computer are vertically integrated and do not leverage either today’s semiconductor manufacturing ecosystem or state-of-the-art classical supercomputing infrastructures. One alternative is a more horizontal advanced development approach based on a consortium across supercomputer integrators, HPC platform and EDA tool developers, and semiconductor fabrication specialists, all guided by quantum computing experts. The consortium’s combined skill sets could speed up the creation of a quantum computer that can solve utility-scale problems by enabling the building blocks and their relationships as shown in the schematic. The mandate of the consortium would be benchmarking the five following key components of the hardware–software stack:

1. **Qubit fabrication** (Section II A) can be developed in a new 300-mm prototype foundry using custom state-of-the-art cluster tools that only exist at this scale. The quality and yield of the qubits could be simultaneously improved. For scaling, new metrology tools should be developed that use standard in-line defect tools to benchmark qubit yield.
2. Scaling the quantum computer to 20k qubits/wafer could use **wafer-scale integration** (Section II B), as already demonstrated by the semiconductor industry for 300-mm wafers. Feasibility benchmarking can use established superconducting and micromachining processes, but be concurrently developed at 300 mm. This design allows all electrical connections to be at 3–4 K, making it much easier to scale.
3. **Control hardware** (Section II C) can be realized and benchmarked by existing room-temperature control electronics for up to a thousand qubits. The 20k-qubit scale could be achieved by a combination of i) wafer-scale integration, ii) high-density cables

and interconnects, iii) a moderate level of time and frequency division multiplexing (1:4 or 1:8), and iv) low-power, high-density digital-to-analog front-end development. Finally, to reach the 1M-qubit scale, dedicated and integrated CMOS could be developed that may operate in cryogenic temperatures. In addition, tight integration between the control hardware and the compute resources of the HPC system must be developed to allow for efficient calibration workflows that optimize and stabilize fidelities while not limiting uptimes and system utilization.

4. For fault-tolerant quantum computing (Section III) an **error-correction decoder** (Section III F) can be tested using an HPC-accelerated FTQC emulator (Sections III B and III E and Appendix E) to synthesize syndrome measurements of QEC codes. This test data could then be directed into the control hardware and used to benchmark the decoding CPU/GPU hardware and software in real-time emulation. For utility-scale FTQC, the decoding software system may require incorporating all ideas of distributed, hierarchical, and moving-window decoding to support 1M+ qubits.
5. The quantum computer can be integrated with conventional HPC infrastructures in six different layers (Section V): **heterogeneous quantum–classical coprocessors**, adaptive circuit knitting, FTQC compilers, distributed decoders, calibration, and control. A dedicated ultra-low latency, real-time QEC network that includes the control hardware, the decoder hardware, and the logical circuit orchestration hardware could be utilized to execute the FTQC workflow.

System integration of the various quantum hardware and software components, particularly as the number of qubits scales, should be tested throughout development based on state-of-the-art metrology technologies and protocols currently used by the HPC industry.

II. TOWARD HIGH-QUALITY QUANTUM HARDWARE AND HIGH-PERFORMANCE CONTROL

It is well understood that a limiting factor to realizing a practical quantum computer is construction of high quality qubits with high performance control. In this section we will discuss how superconducting qubits could be fabricated using the latest semiconductor fabrication techniques, followed by how to leverage the latest innovations in wafer scale integration to connect qubits to a microwave control system. Finally, we end with how the microwave control hardware can be engineered in both a scalable and economical fashion.

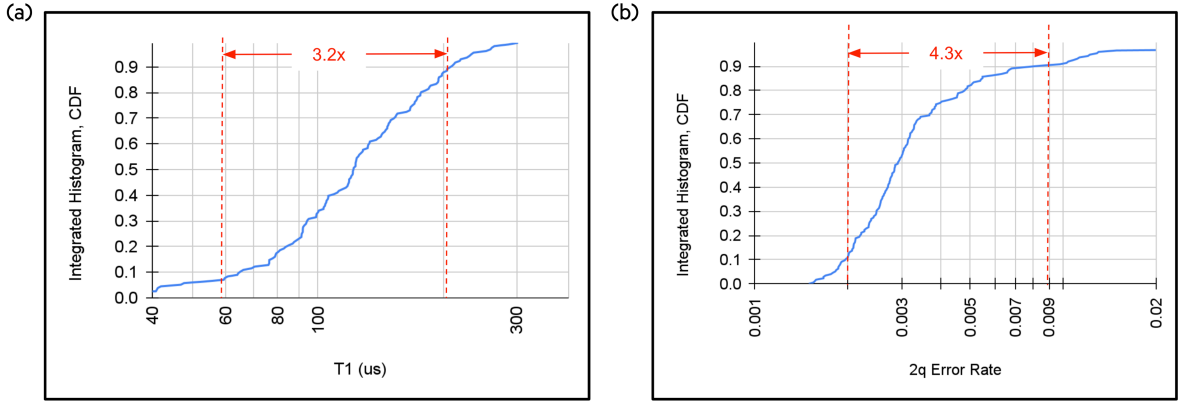


Figure 2: IBM Heron (Fez) device performance taken on 08/06/2024 from calibration data accessible via IBM’s quantum cloud. (a) Distribution of T_1 across 155 qubits. (b) Distribution of two-qubit (2q) error rates across 351 qubit pairs.

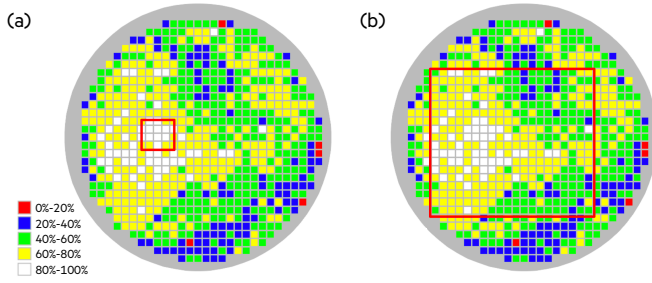


Figure 3: Illustration with simulated data on how the effect of fabrication uniformity and qubit size on median qubit error. (a) When selecting from a small number of qubits, it is possible to cherry pick the best qubits on the wafer and avoid outliers. (b) When selecting from 300+ qubits, it is impossible to avoid fabrication outliers.

A. Qubit fabrication

Researchers often report the performance of their best superconducting qubits, as opposed to more meaningful metrics such as an average, or as more appropriate for systems engineering, the worst device performance. A current challenge is thus to fairly compare approaches and build reliable models for improving coherence. In Table I, we derived the “Target” hardware parameters needed to achieve an FTQC error suppression rate for practical quantum advantage. In particular, we introduce a new “tailedness” metric that describes the distribution in qubit quality. For more information on how these metrics were obtained, refer to Appendix B.

Superconducting qubits have achieved coherence times of $T_1 \gtrsim 100 \mu\text{s}$ and two-qubit errors of 0.1%; however, it is not possible to achieve such results uniformly across a large wafer, as illustrated in Figure 3. A 300-mm wafer can accommodate roughly 20k superconducting flux tunable qubits with adjustable couplers. Furthermore, there is evidence that the two-qubit errors are sensitive not to the average T_1 times, but likely the worst T_1 times across an entire wafer. In Figure 2, we see the T_1 spread of qubits on an IBM Heron processor and the associated spread in two-qubit error rates. A direct correlation be-

tween T_1 and two-qubit error spreads is yet to be properly studied. However, it is known that as the TLS density increases, the harder it becomes to calibrate the device to avoid TLS and the higher the probability a TLS occurs in the adjustable coupler. Therefore, it is imperative to leverage advanced fabrication to improve the uniformity of performance for each qubit in a large wafer.

Based on this challenge, a systems engineering principle that can be used to guide qubit metrology is “the worst 1% of devices determine system performance.” For quantum computers, this is a useful design rule since an $\sim 1\%$ qubit dropout rate for the surface code will cause the system to degrade or fail. Our goal is thus similar to fabricating complex CMOS electronics: make every qubit identically good. Our plan is based on the categories of quality, quantity, speed, and connectivity introduced in the previous section, but organized according to hardware subsystems. Fortunately, we have found that many of the system engineering constraints can be solved concurrently. For example, we will explain how qubits can be fabricated in a manner to simultaneously improve both quality and scaling.

Quality. Both single- and two-qubit error rates are targeted to be in the 10^{-4} range for both NISQ and error-corrected quantum computers. As adjustable couplers have achieved two-qubit error rates in the low 10^{-3} range with modest coherence times of $20 \mu\text{s}$, it should be possible to meet this metric with reasonable $10\times$ improvements in the T_1 coherence time. This coherence requirement has indeed been met with tunable and non-differential qubits made from Al in the academic laboratory of R. McDermott at the University of Wisconsin. Average T_1 times are in the $100\text{--}200 \mu\text{s}$ range, with a “hero” device showing T_1 as long as $800 \mu\text{s}$. This improvement came from identifying a source of TLS defects, then minimizing its contribution in the design and fabrication. One can continue building better coherence models and improving the interface quality. This process looks to be compatible with multiple qubits and adjustable couplers.

Fabrication. We do not assume this academic process is good enough, it only shows that our models and

| Hardware Parameter | Baseline | Target | Desired |
|----------------------------------|-------------|-------------|-------------|
| T_1, T_2 times | 100 μ s | 200 μ s | 340 μ s |
| T_1 tailedness | 71 μ s | 23 μ s | 23 μ s |
| Single-qubit gate error | 0.0004 | 0.0002 | 0.00012 |
| Two-qubit gate error | 0.003 | 0.0005 | 0.00029 |
| State preparation error | 0.02 | 0.01 | 0.00588 |
| Measurement error | 0.01 | 0.005 | 0.00294 |
| Reset error | 0.01 | 0.005 | 0.00294 |
| Single-qubit gate time | 25 ns | 25 ns | 25 ns |
| Two-qubit gate time | 25 ns | 25 ns | 25 ns |
| State preparation time | 1 μ s | 1 μ s | 1 μ s |
| Measurement time | 200 ns | 100 ns | 100 ns |
| Reset time | 200 ns | 100 ns | 100 ns |
| Error suppression rate Λ | 2.34 | 9.3 | 18 |

Table I: Hardware specifications for three sets of parameters: baseline, target, and desired hardware. The baseline set represents the state-of-the-art values; the target set is envisioned to be a promising near-term goal; the desired set of synthetically generated hardware specifications corresponds to a noise model with about twice the error suppression rate, Λ , of the target hardware extracted from the exponential suppression law $\mu d^2 \Lambda^{-(d+1)/2}$ for quantum memory. Our benchmarking studies resulted in $\Lambda \approx 2.34$ for the baseline set and $\Lambda \approx 9.3$ for the target set, respectively. The desired model is specified by the value $\Lambda \approx 18$ and is therefore also referred to as the “ Λ_{18} model” in this paper. The T_1 tailedness characterizes the weight of poor-quality qubits with respect to variations in coherence times across the qubit chip. As discussed in Section IIID, the standard deviation is used as the metric for tailedness in this paper, while the effects of higher moments (such as skewness and kurtosis) can also be crucial given that realistic distributions of T_1 values have significantly heavier tails compared with the associated approximating Gaussian distributions.

fabrication plans for improving qubits are on track. An example is our insights on TLS defects, a significant issue for present-day superconducting qubits. Two-level systems introduce sparse defects at random frequencies which lowers coherence for a significant fraction (1–10%) of qubits, thus they cannot be statistically avoided in large systems. Preliminary data of our fabrication process shows a much lower density of these TLS defects, and even a significant tightening of the spread of T_1 coherence times.

One can correlate decoherence with defects in the qubit fabrication. For example, we calculate that extra loss from TLS will occur with 0.1- μ m-diameter particle defects, which is detectable with in situ optical defect metrology. These in-line tools, which are standard in CMOS processing, can thus act as a proxy for qubit quality and be used to rapidly optimize fabrication provided that one can build an adequate physics-based model.

We believe the necessary improvements in qubit fabrication are only possible using modern semiconductor tools and processes rather than those that are decades old. For example, we should eliminate lift-off, which is easy to use but known to be dirty. Another example is fabricating in modern cluster tools, which allow multiple process steps without breaking vacuum. This minimizes qubit loss coming from amorphous interfaces that are only a few nanometers thick. Figure 4 shows how an in situ cleaning and deposition process for Al on Si, the most critical metal-substrate interface, yields an atomically sharp interface. We also developed a process to improve the substrate-air interface, next in importance.

Applied Materials and Qolab are using tools that improve every process step. Using 300-mm wafers allows access to modern metrology tools to monitor and improve defects and yield. Because Applied Materials builds fab-

rication tools and has a prototyping cleanroom, it is less expensive to modify or retask these expensive cluster tools for this custom quantum process. A key issue preventing progress in the field is that existing groups do not publish their die yields or qubit yields for larger processors (e.g., there is no data on the T_1 time of the IBM Condor 1000+ qubit device). One should collect detailed metrics on die and qubit yield, and correlate the performance to room temperature measurements such as optical metrology and junction resistance spread.

Qubit design. Our fabrication process is designed to be flexible and thus compatible with a variety of qubit designs. We are building adjustable transmon qubits and couplers since it is possible to have both fast gates (30–40 ns) and long coherence times ($> 100 \mu$ s). Ideally, the error per gate is approximately the ratio of the gate to coherence time. Our analysis of the adjustable coupler system indicates that intrinsic control errors (disregarding T_1 and T_2 decoherence) should allow two-qubit errors in the 10^{-4} range. Another aspect of the qubit design is to engineer robustness to gamma and cosmic rays [76] [25].

PDK. A process design kit (PDK) is a central feature in the design of conventional circuit chips. This is generally supplied by the foundry that will produce the chip, and is based on a particular fabrication process supported by the foundry. In short, the PDK provides all of the information a design engineer would need to architect the chip to the specifications required. However, there may be separate third-party libraries or other information which would supplement the materials provided in the PDK.

One could create an analogous PDK for superconducting qubits as one develops the technology as described in this document. Initially, the PDK will provide informa-

tion needed to perform the mask layouts for the initial qubit test chips. The main component will be a technology file for the layout editor. This would define the layers used in fabrication and their purpose. Additionally, device models for the Josephson junctions for use in a circuit simulator, based on parameters measured from the tech chips, will be provided. The PDK would also provide basic physical and electrical information, such as minimum linewidth, minimum spacing, etc. as supplied by the facility performing qubit fabrication. As the designs become more complex, one could add additional descriptions for design rules allowing automated design rule checking (DRC) and circuit connectivity so layout vs. schematic (LVS) testing can be performed. This is entirely analogous to initial stages for PDK development for a standard digital process, a task Synopsys has performed on innumerable occasions.

Note that PDK setup files are dependent on the tools used in the design flow. In some cases, one could support multiple tools that might be in use at different sites within our group. Synopsys has industry-standard tools for layout, DRC, and LVS, and others, which would be brought to bear on the project. As the technology develops further, additional tools more specific to quantum will come into use, and the corresponding technology files will be added to the PDK. For example, unlike in conventional digital circuits, extraction of precise values for capacitance and crosstalk will be needed. This will require the use of a field solver. There will be additional interfaces to specialized software used for modeling and simulating qubits and quantum components at a higher level. The results from the field solver will be back-annotated schematics which can then be simulated to yield results that include parasitic elements, analogous to parasitic extraction of a conventional design.

One can add parameterized cells (PCells) for elements that are used multiple times in our designs. Parameterized cells “draw” themselves (as mask patterns) according to the values of one or more parameters provided. This is for convenience when one needs multiple instances of devices with varying parameters. Most conventional PDKs provide a library of PCells for different types of device supported by the process. It is likely that we will have analogous needs.

Finally, the PDK will provide an area for documentation of all the process steps and other useful information developed along the way as it relates to specific procedures using the supported tools, as well as “golden” results that can be used for comparison purposes.

B. Wafer-scale integration

Present superconducting qubit devices have yield issues even at fifty to a few hundred qubits. As with conventional electronics, the current solution is to dice the wafer into many dies, test the dies, then assemble the working ones into a larger system. This solution is not

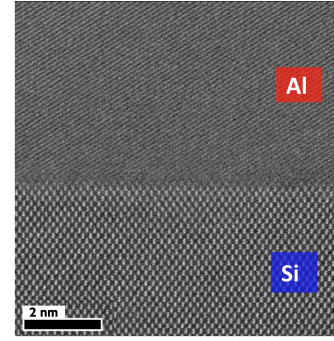


Figure 4: Atomically sharp aluminum-to-silicon substrate interface from Applied Materials’ cluster tools used for qubit fabrication [77].

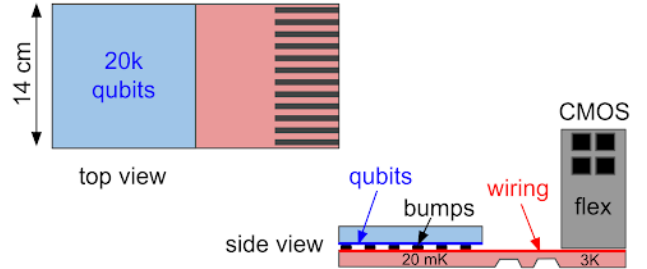


Figure 5: Qubit wafer (blue) bump-bonded to a 300-mm wiring wafer (red). The wiring wafer is thinned by micromachining for thermal isolation between the qubit temperature (20 mK) and 3 K. The wiring wafer connects via spring contacts to flex circuitry (gray) for the control wiring and CMOS electronics.

ideal for qubits due to communication bottlenecks and coherence loss between dies.

A promising solution is fabrication on 300-mm wafers with high-quality processes, which naturally allows for qubit scaling using wafer-scale integration. This concept only works for low defect densities, which we believe is possible for three reasons. First, one should use CMOS-type process and tools that are known to give high yields. Second, the critical area of the qubit devices are many orders of magnitude lower than typical electronic devices, with only a few $0.2\text{-}\mu\text{m}$ -sized junctions/ mm^2 , and critical lithography dimensions typically $\sim 1\text{ }\mu\text{m}$. Third, a process should be developed in a cleanroom that has extensive metrology tools for automatic detection and optimization of defects.

Using the center portion of the wafer, 140 mm by 140 mm, and a 1-mm qubit spacing, the number of qubits per wafer is about 20k. Note these qubit devices can be patterned using conventional deep-UV optical lithography.

Superconducting wiring. One of the most difficult engineering tasks when scaling to a large number of qubits is connecting the qubits to their analog control signals or readout. This escape wiring is especially difficult at the qubit temperature of 20 mK because the wiring is typically made using a shielded transmission line such as coax or, for adjustable qubits with DC connections, expensive superconducting NbTi coax.

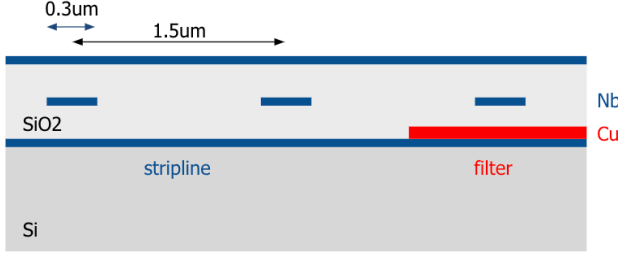


Figure 6: Cross section of wiring wafer, showing stripline width 0.3 μm and pitch 1.5 μm . The right stripline shows design of a low-pass transmission-line filter using a copper damping film.

A solution for scalable wiring is to use wafer-scale integrated-circuit superconducting wiring from 20 mK to the 3–4 K stage, as shown in Figure 5. The qubit chip described previously is indium bump-bonded over its entire wafer to the wiring wafer. The Nb wiring are stripline transmission lines for good isolation and a 20–50 Ω impedance. With a 0.3 μm center width a 1.5 μm pitch, about 92 k wires can be routed across the wafer in a single wiring layer. As shown in Figure 6, transmission-line low-pass filters can be integrated into these wires to be compatible with present-day designs. The wiring layer uses micromachining processing to thin the wafer between the 20 mK and 3 K stages, with multiple thinned sections for connection to intermediate temperature heat sinks.

The fabrication of the wiring wafer assumes low defects on a single wafer, which fortunately requires a relatively simple multilayer metal and insulator processes with vias. Such processes already exist for classical Josephson electronics. The sensitivity to defects in the wiring wafer is clearly higher than for the qubit wafer, but with 0.3- μm -wide wires, modern processing should provide good yields. Over the last few years, there have been significant advances in wafer-scale packaging. In particular, TSMC has developed a wafer-scale integration solution of Cerebras Systems’ AI processors which utilized the entire 300-mm wafer [78], and has developed new packaging solutions for Nvidia’s Blackwell processors [79]. Applied Materials has also developed processes and tools for wafer-to-wafer bonding and heterogeneous integrations [80].

Subsystem modularity. The wiring wafer is connected to a flex circuit board and CMOS control electronics via spring connectors at 3 K. These connections need not be superconducting because the acceptable heat load at 3 K is much higher than at the qubit stage (20 mK). Spring connections allows this qubit+wafer subsystem (Figure 5) to be readily modularized; it can be tested separately then installed in a larger system. This integrated design is useful since this qubit system can be thought as being controlled at 3 K, or virtually at 3 K, with only lower-temperature thermal connections needed to cool the chip.

Measurement and readout. Another scaling bot-

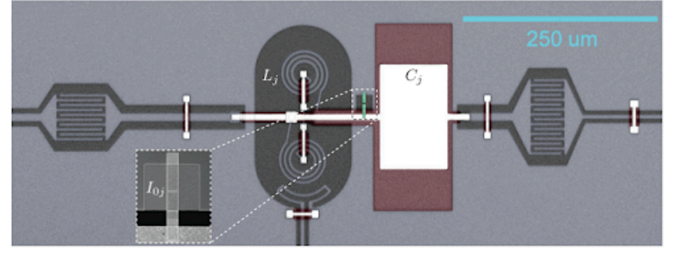


Figure 7: Qubit readout using a Josephson photomultiplier circuit which can be integrated into the qubit wafer. This design eliminates the need for large circulators and parametric amplifiers.

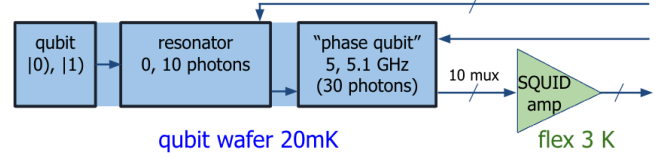


Figure 8: Signal flow for readout. Conventional dispersive readout maps the $|0\rangle$ or $|1\rangle$ state of the qubit to 0 or 10 photons. With 10 photons, a flux-biased phase qubit is driven to change its flux state, which changes its small-signal oscillation frequency from 5.0 to 5.1 GHz. This classical state can be measured with a large number of photons in a multiplexed manner with a low-power SQUID amplifier at 3 K.

tleneck is qubit measurement and readout, as typical systems require circulators and parametric amplifiers that have a volume of $\sim 1 \text{ cm}^3$ or more. Qolab plans to use a readout technology that can be readily integrated into the qubit or wiring wafer, as described in Figs. 7 and 8. Readout using the Josephson photomultiplier, developed by McDermott [81], has achieved acceptable fidelity of 99% and can further be improved. Because the measurement forms a classical state on the integrated detector, it can be readout in a multiplexed manner with a simple superconducting SLUG (SQUID) amplifier at 3 K, without needing circulators. CMOS drivers at 3 K have been developed to bias the large number of SLUG amplifiers.

Scaling through tiling. For quantum computers larger than 20k qubits, one could create baseline system that uses subsystem tiling, as illustrated in Figure 9. Here, each qubit wafer is precisely mounted onto an invar frame so the modules can be mechanically assembled with precise capacitive coupling between the wafers. These “edge couplers” communicate between each tile and can have higher error rates than qubits within each tile.

Figure 9 illustrates linear tiling, but a serpentine pattern is possible for a more compact area. A system with 5–10 tiles can fit into a large dilution refrigerator using

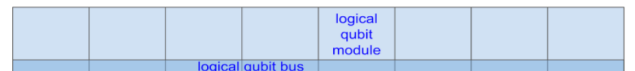


Figure 9: Tiling of multiple qubit subsystems. Each subsystem has 20 k qubits, as described in Figure 5.

present-day designs. Scaling up to many more tiles would clearly require new designs for a large cryostat. Optical connects [82] could provide an alternative solution allowing easier scaling with modular cryostats. In this case, it would be particularly useful to separate the T -gate distillation factory from the main processor. These ideas will be incorporated as soon as optical to microwave quantum transducers are available. Resource estimates for both capacitive interconnects and optical interconnects is described in Section III G

Scaling cross-talk simulations. Numerical electromagnetic (EM) field solvers are currently used to determine electrical parameters for qubit circuits. They work well on a small number of qubits, but become prohibitive at scale due to the difficulty of meshing from the sub-micron film thickness to the centimeter-or-greater size of the chips. Although qubit parameters can be determined well by isolated simulations, the evaluation of crosstalk is particularly difficult since it requires simulation of the entire circuit, including the chip mount.

Running brute-force EM simulation even on a powerful computer will eventually run into the capacity limitation, especially when the number of superconducting qubits grows to 100–1000 for wafer-scale integration. To solve for a large number of superconducting qubit layout geometries with a rigorous numerical EM simulation approach, domain decomposition method (DDM) techniques could offer the ability to use a distributed network of compute nodes and leverage larger blocks of distributed memory. A DDM decomposes a mesh representation of a model into a series of non-overlapping mesh domains that, when each matrix is individually solved with a traditional direct matrix solver, could collectively be used as a preconditioner for an iterative matrix solution to the full model. A generalized scheme, in which a given geometry for simulation is meshed in whole, is developed, resulting in a mesh that is automatically subdivided into equal sized mesh domains for balanced parallel computing. Figure 10 shows an example where a DDM is successfully applied to a 1024-element antenna array.

For an antenna array solved with this general approach, the meshing processes can be quite expensive for the entire array. However, in the approach discussed here, one could leverage the repetitive geometry of an array: only a single unit cell is meshed, then it can be repeated along the array lattice to develop a set of mesh domains for the entire finite-sized array. Each cell of this array will have a unique solution depending on its location, and the resulting full solution takes into account the effects along the edge of the array. The approach is efficient as individual cells can be solved in parallel. Further efficiencies are realized by repeating matrices that result for certain cells residing in identical environments. We believe this technique can be leveraged to solve a superconducting qubit crosstalk simulation with on the order of 1000 qubits.

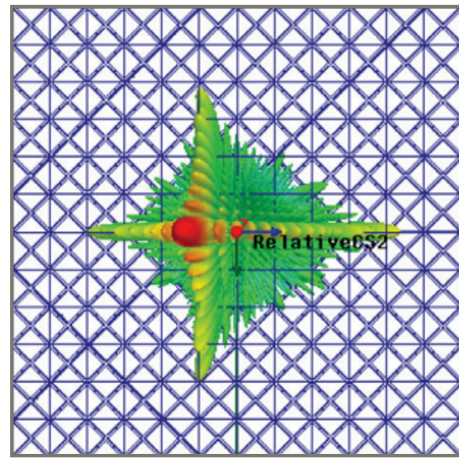


Figure 10: Domain decomposition method (DDM) for a 1024-element antenna array.

C. Control hardware

Engineering a high-performance quantum control system is necessary to manage the quantum processor unit, execute calibration and application sequences, and interface with additional classical compute resources such as CPU/GPU servers. Such a control system is responsible for:

1. Generating and orchestrating the precise pulses that drive the quantum system dynamics
2. Reading out qubit states (including digital signal processing and state discrimination)
3. Processing data and making real-time decisions, including conditional operations, control flow, and control operation parameters updates
4. Integrating with additional classical resources
5. Providing suitable SW interfaces for productive development and for integration with SW components that are higher level in the stack

Generating pulses within the coherence time of the qubits, acquiring qubit measurements, and processing these measurements for real-time feedback and efficient data transfer requires a unique digital processor architecture, which we refer to as the pulse processor unit. Then, the system's analog front-end, which includes the digital-to-analog and analog-to-digital converters as well as the analog signal chains (amplifiers, filters, attenuators, etc.), generates and acquires analog signals. As the system scales, maintaining analog performance, particularly in terms of noise, stability, and cross-talk, becomes crucial for achieving the fidelity targets necessary for NISQ computing or QEC.

Scaling quantum control. With current qubit quality and scale up to 1k qubits, the control system architecture should focus on performance and flexibility. Such

performance and flexibility is needed so the control system does not limit overall performance or the speed of testing, research, and development iterations, even at the cost of overall design. As the system scales, the density of control electronics and wiring presents challenges in terms of space and heat dissipation. Moreover, cost per qubit control becomes a significant issue. Hence, once the desired fidelities are achieved and the requirements for errors and scale-up are well understood, some control system requirements can be relaxed. This would allow optimizing for cost, power consumption, and size while maintaining target fidelities. To reach 20k qubits, one could employ a combination of low-power, high-density, room temperature analog front-end development in conjunction with cryo-CMOS components for moderate 1:4, or 1:8 frequency and/or time division multiplexing. To reach 1M qubits, one could develop dedicated, integrated CMOS that may operate in cryogenic temperatures. Effective co-design of room-temperature and cryogenic control electronics will be critical for seamless integration as the system scales.

Another possibility for scalable qubit control might arise from our advanced fabrication, assuming an improvement in qubit quality and reproducibility: if the qubits can be fabricated nearly identically and with a low probability of TLS dropouts, then a single control signal can be split to many qubits, with simple variable amplitude and phase adjusters at 3 K to fine-tune signals. This will enable an integrated and low dissipation solution to control.

Room temperature and cryogenic control. As quantum systems scale, the benefits of cryogenic control increases. Wafer-scale integration enables the connection of flex cables at 3 K, where a maximum heat load of 41 W is expected to be sufficient for flex cables connecting control at 77 K or 300 K. We identify several trade-offs between room-temperature and cryogenic control at 77 K or 3 K:

1. Power consumption: Operating at cryogenic temperatures with lower V_{dd} and reduced signal amplitude, proportional to the reduced thermal noise, leads to significantly lower power consumption—potentially orders of magnitude lower compared to room-temperature systems.
2. Size and system complexity: Cryogenic control, such as at 77 K, results in a more compact system, housed entirely within the cryogenic refrigerator. This not only eliminates the need for multiple racks but also reduces the number of flex cables required between the cryogenic and room-temperature stages, leading to a more efficient setup.
3. Control functionality and flexibility: Room-temperature control benefits from the flexibility of incorporating additional computational resources, such as logic circuits, filters, and classical computing, with the option to increase rack space

as needed. In contrast, cryogenic systems may face limitations in computational capacity, though emerging cryogenic-compatible technologies could alleviate this.

4. Process and cost: Room-temperature electronics utilize established CMOS processes, benefiting from mature manufacturing ecosystems, stability, and relatively low costs. Cryogenic control, however, requires more specialized processes like fully depleted silicon-on-insulator (FDSOI), optimized for low power. Broader adoption of these processes in industries like aerospace could drive down costs and improve availability, facilitating wider adoption.
5. Signal noise and stability: Operating at lower and more stable temperatures may reduce noise and drift of the signal properties. The reduced thermal noise allows for lower signal amplitudes, potentially reducing the complexity of amplification stages. On the other hand, the limited space and power in the cryogenic control introduces challenges such as crosstalk, noise introduced by analog up-conversion required to avoid high DAC clocks and limited power budget for amplification.

Based on the current analysis, low-power digital-to-analog converters (DAC) along with their analog chains are crucial components for enabling low-temperature control. Achieving a power target below 1 mW per channel could enable overall analog and digital power consumption of approximately 100 W at 77 K, sufficient for controlling a 20k qubit system. Once the target qubit quality is achieved, a comprehensive end-to-end trade-off analysis should be conducted to determine the optimal architecture for practical implementations, considering both cryogenic and room-temperature control.

For systems of 10k–100k qubits, local optimization of the QPU, packaging, room-temperature control, and cryogenic electronics become inadequate. Instead, the control system must be optimized as part of a holistic solution. For instance, channel crosstalk and pre-distortion issues can be addressed either at the QPU and packaging level or within the control system. A localized optimization would yield suboptimal results, whereas a co-design approach can produce an optimal solution. Control filters should be tailored to the package and QPU channels, while crosstalk can be mitigated at the CPU and package level, ensuring the control architecture is aligned to correct for the residual errors. Similarly, analog characteristics of the control system such as phase stability should be optimized based on gate implementation to meet the system’s architectural requirements and prevent unnecessary complexity and cost.

In large-scale systems, both control systems and quantum processors are prone to failures. Consequently, it is essential to incorporate mechanisms for testing functionality at both the system and component levels. Decoder

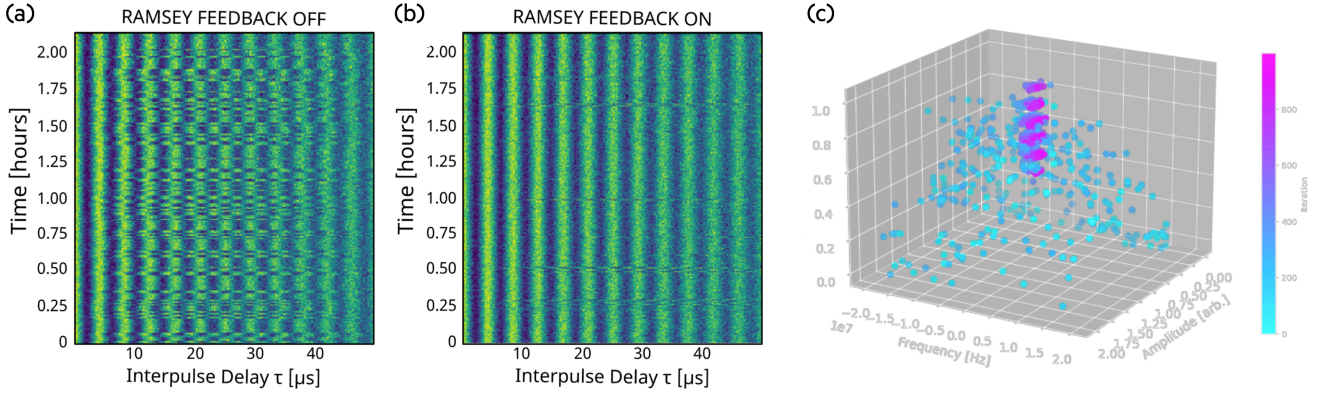


Figure 11: (a–b) Example demonstrating the impact of frequent calibration, showing Ramsey scans over time performed without and with real-time tracking of the qubit frequency [83]. (c) II-pulse amplitude and frequency 2D optimization demonstrated on a DGX Quantum system.

results from benchmark circuits can offer insight into the overall operational status of the system. Additionally, benchmarks focusing on one or few surfaces provide a means to diagnose localized failures. Benchmarks evaluating the fidelity of physical qubits can offer insights into qubit calibration status as well as potential hardware issues. Furthermore, component-specific benchmarks such as those for amplifiers are necessary to isolate and identify exact points of failure. These benchmarks should be optimized for execution in minimal time to enable frequent testing, and may also be used to trigger calibration procedures when necessary. Because large-scale control hardware is susceptible to local errors, robust management and monitoring features—encompassing telemetry and self-testing—are required to detect, diagnose, and address such issues effectively. Upon detection of an error, the logical circuit orchestrator is notified, ensuring the platform continues to function while debugging and calibration processes are applied to resolve the faults.

To facilitate debugging and hardware updates while maintaining an operational system, the control platform is structured with multiple clusters. Each cluster is managed independently while remaining synchronized with the others, thus enabling the isolation of individual clusters for maintenance or error handling without disrupting the broader system.

Control system integration with classical compute resources. By their nature, quantum computing workloads require quantum–classical iterations. This includes calibration workflows, NISQ hybrid applications, and quantum error correction. Quantum–classical iterations require a low-latency, high-throughput interface with the right division of responsibilities that allows transferring a compact representation of the data across systems. For applications that require significant compute resources, tight integration to HPC clusters is required to efficiently utilize the quantum and classical resources.

Calibration impacts qubit fidelity and plays a critical role in determining the error correction code distance, which in turn directly affects the scalability and com-

plexity of the decoding process. A significant challenge is variability in qubit fidelity; the system’s overall performance is often constrained by the lowest-performing outlier qubits. To maximize the performance of these outliers, advanced calibration strategies such as optimal control, reinforcement learning, demonstrated in Figure 11, and model-based simulations may be used. To meet the fidelity goals for large quantum processors, it will be necessary to have scalable and efficient calibration routines that enable concurrent calibration across qubits. Reducing the calibration time allows for repeating the calibration more frequently and allocating a larger portion of the time to optimizing outlier qubit performance. Figure 11 demonstrates the fidelity improvement from frequent calibration, tracking the drift in a qubit frequency. A key requirement for calibration flows is minimal execution overhead and feedback latency. A typical calibration node requires executing thousands or more shots. A shot is structured with initialization, executing a pulse sequence, and measurement, and typically takes hundreds of nanoseconds. As such, the calibration routine overhead should be minimized accordingly to few milliseconds from loading to gathering the measurements statistics.

Architecture of a quantum–classical integrated platform for FTQC. FTQC workflows require a quantum–classical integrated platform to execute the logical circuit, orchestrate QEC decoding and surface operations, and control physical qubits. The main components of the platform illustrated in Figure 12 include the quantum control, QEC acceleration servers, and dilution fridges that hold the quantum processor and cryogenic control. The quantum control is divided into clusters; typically each cluster controls a set of surfaces, and the independent clusters simplify the control and provide resiliency to errors while maintaining synchronized execution. The QEC acceleration servers are responsible for real-time decoding and logical circuit orchestration. The servers leverage general-purpose GPU/CPU systems with the option for additional ASIC or FPGA accelerators. The low latency QEC network provides an efficient interface for transferring syndromes and surface

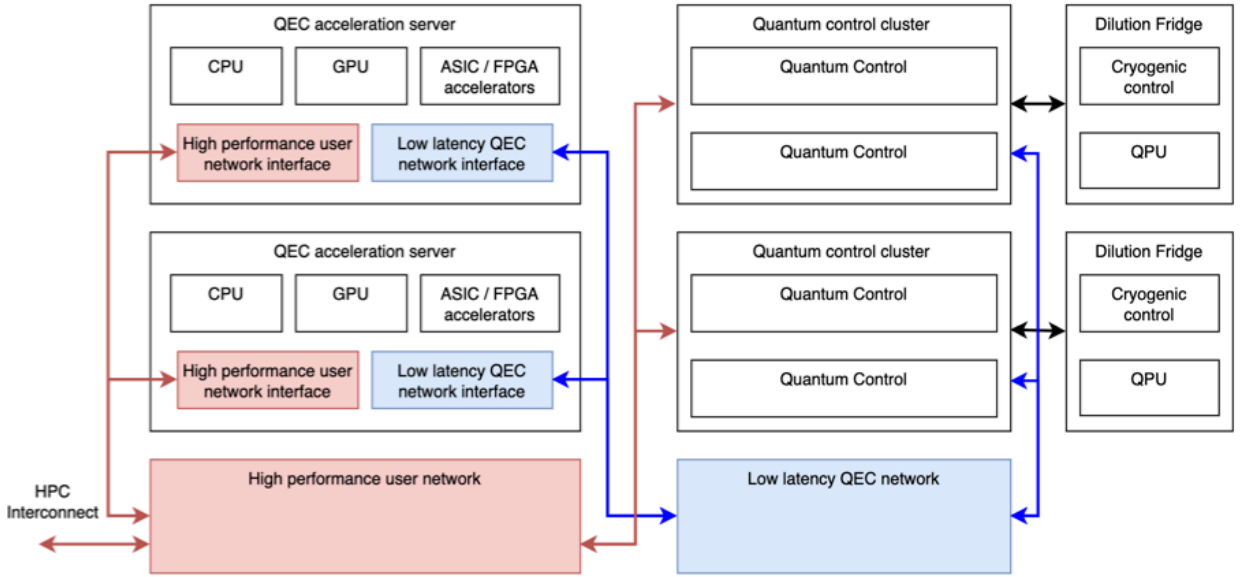


Figure 12: Illustration of an FTQC platform. Multiple quantum control clusters are connected to multiple QPUs. The platform includes optional cryogenic control located adjacent to the QPU. The quantum low-latency network connects the clusters and acceleration servers that run the QEC decoders, logical circuit orchestration and calibration and optimal control routines. A high-performance user network connects the control and servers and is used for tight integration with HPC compute resources.

operations. A high-performance user network is used for data and program loading as well as connectivity to HPC resources required, for example, for logical circuit compilation.

Minimizing quantum–classical feedback latency. The interaction between the quantum processor and classical control systems is characterized by quantum–classical feedback loops, which can be grouped into three primary timescales [83]. The first is the quantum real-time (QRT) feedback loop, executed within the coherence time of the qubits. An example of QRT feedback is active state preparation, where rapid response is critical to maintain qubit fidelity. Minimizing QRT feedback latency is essential because it directly influences the fidelity and overall performance of the quantum computer. The second feedback loop is the system real-time (SRT) loop, which occurs over one or more iterations beyond the qubit coherence time but is still dictated by the physical system. The SRT loop can be used to track qubit parameter drifts over time, and its speed impacts the system’s ability to adapt to noise and environmental fluctuations, thus affecting the quantum computer’s long-term fidelity. Lastly, the near real-time (NRT) feedback loop is used for classical-quantum interactions that are not constrained by immediate physical properties, but rather determined by the convergence time of quantum algorithms and calibration routines. NRT feedback often requires intensive classical computations and is typically carried out on the application level, outside the quantum control system. Efficient implementation of all these feedback loops is vital for both NISQ and QEC applications.

Integration with high-performance computing

resources. In addition to dedicated classical resources for tasks such as calibration, optimal control, and QEC decoding, the quantum workload may require high-performance compute resources. Examples for HPC use-cases include simulations and modeling, optimal control, and circuit knitting. Efficient integration with HPC requires a low-latency, high-throughput connectivity to HPC systems based on HPC standard interconnects such as Infiniband or Cray’s Slingshot or a dedicated protocol. Next, The control system should also be designed to process data locally, transmitting compact data streams to HPC servers and receiving compact instructions to maximize efficiency.

III. TOWARD FAULT-TOLERANT QUANTUM COMPUTATION

To reliably execute quantum algorithms at utility scale, they must be implemented using nearly noise-free logical qubits, with logical error rates far below the physical error rates of qubits and gates of the QPU. To this end, QEC codes are leveraged to combine many low-fidelity physical qubits into fewer high-fidelity logical qubits [84]. Since the logical quantum state of computation must not be observed during computation, QEC relies on measurements of ancilla qubits to detect the most-probable errors afflicting the code. This introduces additional circuitry to be executed which on its own creates further opportunities for error events. The goal of FTQC is to utilize QECCs in such a way that the rate of production of errors is suppressed by the rate of their correction [85, 86]. Fortunately, the *threshold theorem* guarantees that the over-

head of FTQC scales as $\text{polylog}(1/\epsilon)$ with respect to the desired precision ϵ for computation when the probability of physical erroneous events is below a certain *accuracy threshold*. This was shown by observing that the probability of undetectable errors is exponentially suppressed if the QECC is concatenated iteratively with itself below threshold [12–14].

For superconducting qubits, which are constrained to a 2D topology and nearest-neighbor interactions, a promising family of QECCs is the topological QEC codes [23, 87] in two dimensions, such as surface codes [88] or color codes [89, 90]. Interestingly, for these codes the accuracy threshold is identical to the order–disorder phase transition critical point of certain classical Hamiltonians with quenched disorder [87, 91, 92]. For the surface code, this Hamiltonian is the 2D random-bond Ising model. Therefore, the threshold can be calculated using Monte Carlo simulations of the code at large sizes (i.e., at large distances). Below threshold, increasing the code distance exponentially suppresses the chance of undetectable errors, which allows us to quantify the performance of the QECC using a single parameter, Λ , representing the rate of this exponential error suppression [24, 26].

In this paper, we discuss FTQC architectures based on the rotated surface code [93, 94], a $[[d^2, 1, d]]$ stabilizer code with physical qubits arranged on a 2D lattice, as illustrated in Figure 13 for distance $d = 3$. However, our analyses can be easily adapted to other types of topological 2D codes. The rotated surface code consists of d^2 physical data qubits located on the vertices of a 2D square lattice and $d^2 - 1$ ancillary qubits located inside

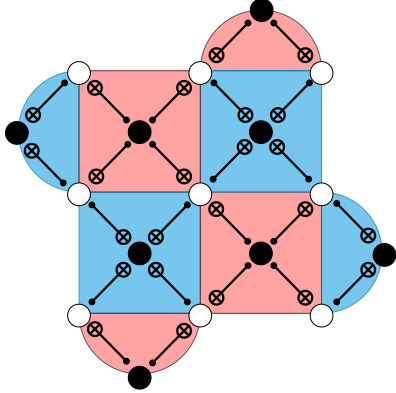


Figure 13: Illustration of the rotated surface code of distance $d = 3$. It uses $d^2 = 9$ physical data qubits (located at the circles of the lattice filled in white) to encode one logical qubit. In addition, it uses 8 ancillary syndrome qubits (located at the circles of the lattice filled in black) to measure the stabilizers. There are two types of syndrome qubits, used for measuring the different types of stabilizers: the ones located in the blue faces measure the Z -type stabilizers, and the ones in the red faces measure the X -type stabilizers. The controlled-NOT gate symbol between the circles filled in white and those filled in black indicate the local physical interactions between the data qubits and neighbouring ancilla qubits used to implement the measurements of the stabilizer generators. The outcomes of these measurements (also called parity checks) can be used to detect and correct a single error on any data qubit in the patch.

different types of plaquettes (depicted as red and blue faces) of an alternating checkerboard pattern within the lattice. The total number of physical qubits needed to implement the code of distance d is thus $2d^2 - 1$.

There are two types of syndrome qubits that are used for measuring the different types of stabilizers associated with the adjacent data qubits. Those located in the blue (red) faces measure the weight-four stabilizers $Z^{\otimes 4}$ ($X^{\otimes 4}$) within the bulk and weight-two stabilizers $Z^{\otimes 2}$ ($X^{\otimes 2}$) on the boundaries. Therefore, all stabilizer generators have a weight of either two or four regardless of the size of the lattice. The error-free logical qubit state is a superposition of the joint eigenstates corresponding to the eigenvalue $+1$ of all the code stabilizer generators. Only local physical interactions between the data qubits and the neighboring ancilla qubits are needed to implement the measurements of the stabilizer generators. The outcomes of these measurements (also called Z -type and X -type parity checks) can be used to detect errors of weight at most $\frac{d-1}{2}$ across the code patch. The logical X (Z) gate can be realized by chains of Pauli- X ($-Z$) operators with boundaries on the top and bottom (left and right) edges.

Since universal quantum computation cannot be realized solely by executing transversal gates on a single QEC code [95], a well-established technique for achieving universality is to implement non-transversal gates by consuming resource states, commonly referred to as magic states, that are distilled with sufficiently high fidelity in magic state distillation factories. In particular, for QECCs with transversal Clifford gates, a non-Clifford gate is implemented by preparing and consuming resource states, such as $|T\rangle = (|0\rangle + e^{i\pi/4}|1\rangle) / \sqrt{2}$ for the case of the T gate. Such magic states can then be used to implement any multi-qubit $\pi/8$ rotation $\exp(-i\pi P/8)$ for $P \in \{I, X, Y, Z\}^{\otimes n}$ acting on an n -qubit system [28]. The creation of high-fidelity logical magic states is an expensive procedure requiring a protocol for their preparation [96–98] that first produces low-quality, low-distance logical magic states, followed by several stages of magic state distillation units (along with code growth steps between them), each of which filters many noisy magic states of low quality into fewer magic states of higher quality.

For the surface code, comprehensive techniques have been developed to perform universal quantum computation. Central among these techniques is *lattice surgery* [94], a method for performing multi-qubit operations on topological QECCs. By performing only physically local operations, the collection of physical qubits comprising different logical qubits (called patches) are merged and split to realize any desired logical operation, where long-range entanglement is facilitated via the use of auxiliary topological patches. Any quantum computation can be compiled down to a scheduled sequence of lattice surgeries [28, 51]. However, to implement non-Clifford gates, the lattice surgeries typically require the consumption of high-quality magic states. A continual

supply of high-quality magic states is essential for this scheme. As described above, these are produced with a certain rate in a magic state factory (MSF) which generates a few high-quality magic states from many noisy ones. The continual production and consumption of magic states requires optimizing the various trade-offs between the space and time costs needed to execute large-scale quantum circuits [50]. During this entire process, any logical data qubits not being acted upon need to be preserved using a quantum memory protocol.

When the code is used as quantum memory, after the projective measurement of all the syndrome qubits in the lattice, the logical quantum state associated with all the data qubits is either stabilized or mapped into a different code word that can be tracked in software by updating the Pauli frame. In contrast, when the lattice surgery is implementing a non-Clifford gate, such a passive error correction strategy cannot be used. In this case, the overhead of decoding and implementing real-time feedback becomes consequential for FTQC compilation.

In what follows, we describe a comprehensive framework for FTQC compilation and execution based on a concept 2D surface code architectures. Our aim is to provide insights as to how the performance of such architectures can be affected by various sources of physical noise, and how improvements in quantum hardware can enhance the performance. In particular, we analyze the performance of various FTQC protocols for several specifications of quantum hardware. These benchmarks are then used for our quantum resource estimation (QRE) studies, presented in Section IV. Our benchmarking studies include additional analyses addressing various open questions. Specifically, we analyze the sensitivity of the performance of quantum memory to different subsets of hardware noise parameters. We also investigate the impact of QPU fabrication process variability (i.e., the tailedness of coherence time and error distributions) on logical infidelity. We then discuss a promising platform for realizing high-performance real-time decoding. Finally, we discuss distributed FTQC involving a quantum network of QPUs, and demonstrate the robustness of lattice surgeries spread among separate capacitively coupled QPU wafers or even separate dilution refrigerators, assuming access to as many weak interconnects as the code distance of the surgery.

A. Fault-tolerant circuit compilation

Fault-tolerant compilation of quantum algorithms is more complicated than that of classical computer programs because the final physical circuit depends on the specific noise characteristics of the quantum processor. It is commonly understood that the number of non-Clifford logical operations (e.g., the T count of the algorithm) is a good indicator of the approximate cost of running the quantum algorithm. However, assembling the quantum program for exact physical circuits to run in hours, days,

or even months on a quantum computer with millions of qubits and sophisticated coprocessors for control and decoding is much more involved.

An operating system for a fault-tolerant computer must therefore perform offline and real-time QPU and decoder characterization, modelling, and performance analysis, and incorporate this information into the compilation pipeline for FTQC execution. We consider three main modules for such a software suite, namely, the FTQC compiler, the emulator (including a noise profiler), and the assembler, details of which are discussed in the following section and summarized in Figure 1. An example of such a software suite, used to conduct our benchmarking studies in this work, is IQBit's TopQAD (Topological Quantum Architecture Design) toolkit [50, 51, 99].

At the highest level of abstraction, an FTQC compiler is responsible for circuit transpilation, decomposition, and parallelization of multi-qubit lattice surgeries on logical data qubits [50, 51, 99]. At the lowest level, the emulator receives noise models from qubit arrays provided by various QCVV experiments to emulate fault-tolerant protocols at lower distances ($d < 30$ typically) and extrapolates logical error rates to higher distances (sometimes 100 or more, depending on the algorithm; see Table IV). The results from the compiler and emulator are provided to the assembler, which is responsible for allocating various zones within the architecture's layout (e.g., for magic state preparations at lower distances, distillation factories at increasing distances, and code growth and switching) and placement of logical qubits in the algorithm zone and scheduling lattice surgeries.

A basic schematic of such a modular quantum architecture layout is presented in Figure 14. In this example, a core processor containing 18 data qubits used to process the algorithmic data is distributed across nine two-tile, two-qubit patches. The core processor also contains a buffer register that allows performing auto-corrected $\pi/8$ rotations by simultaneously connecting the data qubits to a magic state storage qubit and the storage to an ancillary qubit initialized in a $|0\rangle$ state using lattice surgery. The core is connected to a multi-level MSF where the high-fidelity magic states that are consumed in the core are distilled. In the MSF, magic states are first prepared using dedicated preparation units following a magic state preparation protocol. These lower-fidelity magic states are consumed by distillation units to produce higher-fidelity ones in the distilling port. The layout depicted for the distillation units is an example of a feasible layout for the most commonly studied 15-to-1 distillation protocol [100], where 15 lower-fidelity magic states are consumed to produce one higher-fidelity magic state at each distillation cycle. Distillation is conducted in a designated zone, with a sufficient number of distillation units placed side-by-side to facilitate parallel distillation processes, ensuring a continuous supply of magic states between different levels. Once prepared in the distilling ports, the magic states are teleported to a space reserved between levels for expanding the code distance of the

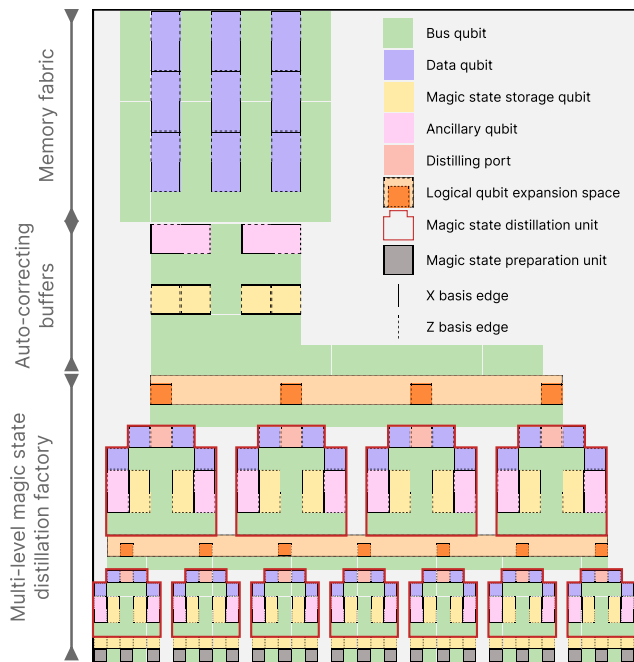


Figure 14: Example of a logical layout of a modular fault-tolerant quantum architecture, of the sort designed by TopQAD [50, 51, 99]. In this example, two distillation levels are used, composed of four and two distillation units each from lowest to highest.

magic states since different qubit encodings can be used throughout the architecture. This process repeats until magic states with the required fidelity are produced at the highest level and sent to the core processor, where they are consumed.

Eventually, the procedures performed by the operating system, including compilation, emulation, and assembly, deliver the exact sequence of instructions for all the stabilizer measurement rounds, logical operator measurements, and conditional recovery operations to be performed by the 1–10M+ physical qubits system to the controller. This information is also provided to the decoder, since it must keep track of the logical protocols being executed (e.g., memory, teleportation, or code growth). We use this framework to conduct detailed resource estimations as presented in Section IV for real-world quantum chemistry problems as well. Furthermore, we study the sensitivity of the performance of the fault-tolerant quantum computer to various hardware parameters in Section III C, which is helpful for guiding the design and fabrication of QPUs.

Resource estimation analyses discussed in Section IV also provide profiles of all the independent lattice surgeries required to be performed on the concept architecture illustrated in Figure 14, and described in more detail in Refs. [50, 51, 99] and also in the appendix. Figure 15 shows an example histogram illustrating the sheer scale of independent decoding problems that must be solved by the decoders. The enormous problem sizes and decoding speed required for a successful execution of FTQC demands a tightly integrated high-performance decod-

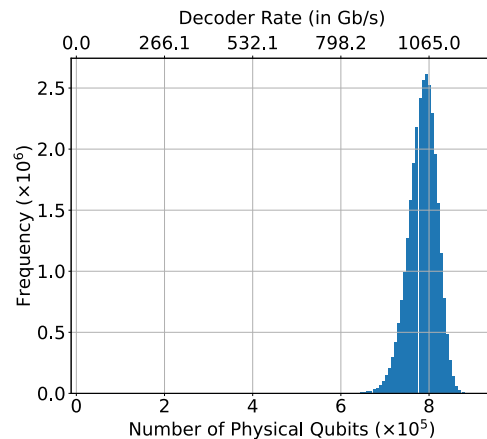


Figure 15: Decoder requirements for electronic-structure quantum simulations of the *p*-benzynes molecule for an active space involving 26 6-31G basis set orbitals, using Trotterization based on the second-order Trotter–Suzuki product formula, with rigorous analytic error bounds (see Section IV). The histogram illustrates the scale of independent lattice surgery procedures that must be performed within the memory zone of the studied topological architecture to execute the quantum simulation circuits, with the top horizontal axis displaying the size of the independent decoding problems that must be solved by decoders. Independent decoding tasks require processing terabytes-large decoding graphs per second. Moreover, independent surgeries can involve 1M+ qubits spread across tens of DRs.

ing system. We describe such a decoding system in Section III F.

B. Benchmarking quantum hardware for the quantum memory experiment

We aim to evaluate how improved physical qubits and gates affect the efficiency of QEC and, consequently, the overall resource requirements of FTQC at a scale of practical use. We focus on 2D lattice surgery using rotated surface codes as our FTQC scheme, although the techniques we developed to this end are applicable to other types of 2D topological QEC codes as well. In what follows, we describe how quantum hardware can be characterized with respect to its performance in realizing FTQC. We do this by emulating fault-tolerant protocols using well-established methods to model quantum hardware noise.

Our benchmarking analyses include three parts. In this section, we show how improved quantum hardware quality affects the efficiency of QEC in suppressing errors. In Section III C, we present the results of a sensitivity analysis investigating which hardware parameters are expected to have the most-significant impact on the performance of FTQC. In Section IV, we demonstrate to what extent improvements in the quality of quantum hardware affect the overall resource requirements of FTQC at utility scale. These benchmarking studies were conducted using the TopQAD toolkit [99].

For the purpose of these analyses, we compare three sets of hardware parameter specifications, which are summarized in Table I: the *baseline* set, which is considered

the state of the art for superconducting qubit technologies; the *target* set representing an achievable near-term goal; and a set of synthetically generated specifications that corresponds to a *desired* hardware model associated with the value $\Lambda \approx 18$. The parameter Λ represents the asymptotic error suppression rate when increasing the code distance by 2 (introduced in Refs. [24, 101] to characterize the QEC performance of FTQC schemes). For these three sets of hardware specifications, in benchmarking the QEC performance for the baseline and target hardware specifications, we obtain the values $\Lambda \approx 2.34$ and $\Lambda \approx 9.3$, respectively, as discussed below. The motivation for considering a desired hardware model yielding the value $\Lambda \approx 18$ is that it is approximately twice as effective as the value of the target set in suppressing errors.

We conduct Clifford circuit simulations to emulate the fault-tolerant protocols required for performing FTQC. The simplest such protocol is the *quantum memory* experiment, which involves only iterative rounds of stabilizer measurements in a single rotated surface code patch representing the fault-tolerant idling of a logical qubit. For this purpose, we employ two open source libraries: Stim [102] for simulating stabilizer circuits and PyMatching for decoding using the minimum-weight perfect matching (MWPM) algorithm [103]. The emulation of other FTQC protocols, such as magic state preparation and teleportation, that are required for a fault-tolerant implementation of an actual quantum algorithm are discussed in Section III E.

We implement a prototypical quantum memory experiment by setting the number of parity-check circuit rounds to match the code distance. Gate and qubit errors are modelled using circuit-level noise with idling errors. *Active noise channels* are applied to the qubits participating in a gate while *idling noise channels* are applied to qubits not engaged in a gate. A brief description of the circuit-level noise model is provided in Appendix E.

Preparation, measurement, and reset gates are executed in the Z-basis with single-qubit Pauli-X channels used to model their errors. Hadamard and CNOT gate errors are modelled using single- and two-qubit depolarizing noise channels, respectively. Single-qubit depolarizing channels are used as idling noise channels. Parameters of the noise channels are determined based on the reference hardware parameters, specifically, by matching the fidelity of the noise channel and the corresponding gate. For active noise channels, the fidelity of the corresponding gate is obtained. For idling noise channels, the target fidelity is that of the dephasing noise channels determined by the concurrent gate duration and the T_1 and T_2 parameters.

The results of our simulations are illustrated in Figure 16. We use numerical simulations at lower distances and extrapolate the logical infidelities at the higher distances in the regime of interest for utility-scale FTQC. To model an exponential logical error decay model, the extrapolation is based the model $\mu d^2 \Lambda^{-(d+1)/2}$, where d is the code distance and μ and Λ are fitting param-

eters. We refer the reader to Ref. [50] for further details on the choice of this error suppression model. We note that previous works instead use the model $\mu \Lambda^{-(d+1)/2}$ to demonstrate an exponential suppression in the surface code error rates *per cycle*. Converting this per-cycle measure to an error model for the entire fault-tolerant protocol results in the model $\mu d \Lambda^{-(d+1)/2}$, which has a linear coefficient d that is different from the coefficient we choose [24, 26, 104]. To mitigate the bias introduced by small distances, data points with a logical infidelity below $10^{-2.5}$ are ignored in the fitting. The extracted Λ value is an important hardware characteristic, as it determines the rate of logical error suppression with distance [101]. We note that the extracted Λ value for baseline and target hardware parameters are, respectively, 2.34(1) and 9.3(3), showing an improvement by roughly a factor of 4. The extracted value of Λ for the Λ_{18} model is 18(1), demonstrating an additional improvement factor of 2 in the error suppression rate as compared to the target parameter set.

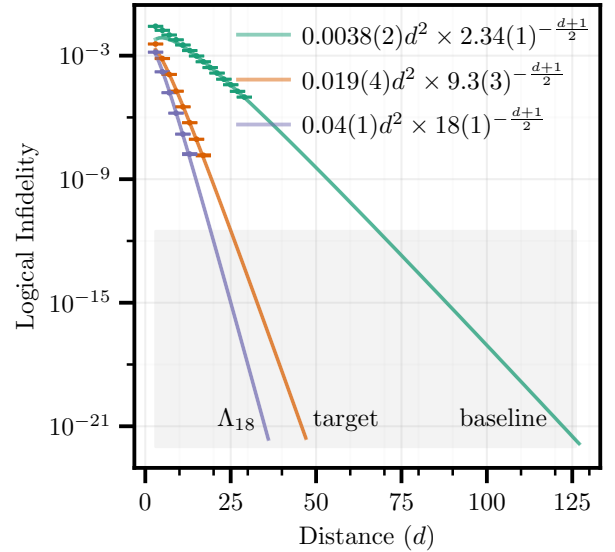


Figure 16: Hardware parameters benchmarked using a logical memory experiment. The numerical dataset is plotted alongside extrapolations derived from a two-parameter fit to $\mu d^2 \Lambda^{-(d+1)/2}$, where d is the code distance. The dataset is obtained using Clifford simulations based on the noise model described in this section. The extracted parameters, along with the fitting errors, are specified in the legend. The shaded region represents the regime of logical infidelities required to implement the electronic-structure simulations' quantum circuits used in our QRE studies, presented in Section IV.

C. Sensitivity of FTQC performance to specific hardware improvements

Quantum hardware engineers often face the uncertainty of which types of noise and errors have the most significant impact on the performance of FTQC, that is, of which hardware parameters are the most critical for achieving improved logical performance. It is unclear

whether the coherence time of qubits or the two-qubit error rates matter most, or the state preparation and measurement (SPAM) errors are most crucial. In this section, we report results of a sensitivity analysis addressing this uncertainty.

We investigate the performance sensitivity of FTQC to specific hardware characteristics, as specified in Table I, by assessing the logical error suppression factor Λ as a function of individual hardware parameters. We analyze the following categories of hardware parameter improvements in the operations involved in implementing the quantum memory experiment: (i) coherence improvements (for idling physical qubits) involving T_1 and T_2 times; (ii) gate-control improvements affecting the Hadamard and CNOT gate infidelities; (iii) SPAM improvements concerning preparation, measurement, and reset errors; and (iv) a combined class encompassing all three groups. We determine the improvement in the error suppression rate Λ when each of these parameter sets are improved separately, while keeping the others constant, as well as when all hardware parameters are improved simultaneously. Our findings are illustrated in Figure 17. We observe that improvements in the gate-control errors yield the most significant impact, whereas improvements in SPAM errors and coherence enhancements are significantly less effective for achieving higher Λ values. The Λ value increase resulting from improving all parameters simultaneously is higher than the sum of individual increments of Λ . Our findings suggest that quantum gate fidelity improvements are more important than SPAM and idling qubit error rates for achieving greater logical performance.

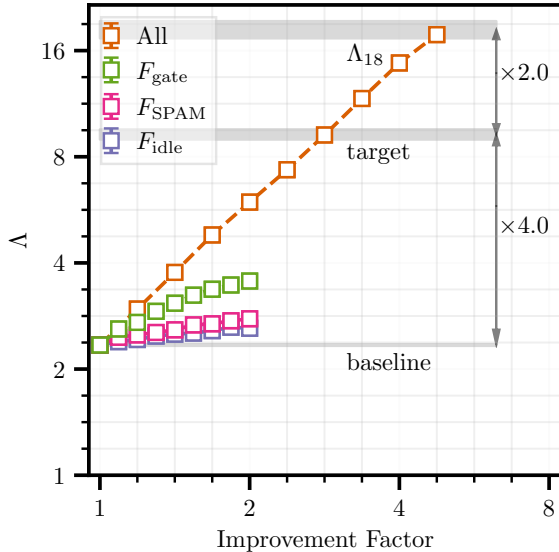


Figure 17: Sensitivity of logical infidelity to hardware improvements. The extracted fit parameter Λ , representing the asymptotic error suppression rate for a quantum memory experiment, varies with a multiplicative improvement factor used to scale a particular subset of physical parameters, indicated in the legend, relative to the baseline hardware parameters.

D. Impact of qubit and gate quality distributions on logical error rates

As our QRE studies in Section IV show, a utility-scale quantum computer is expected to require millions of qubits. Any manufacturing process that produces such large QPUs, or clusters of QPUs, will inevitably create qubits and gates of varying quality. In this section, we present analyses of the possible impacts of process variability on the logical error rates of the rotated surface code.

In order to obtain a realistic distribution of qubit and gate qualities, we use publicly available calibration data obtained for the `ibm_torino` quantum processor [105]. In particular, we focus on the distributions of the T_1 times, as well as single-qubit, two-qubit, and readout errors. In Figure 18 we plot the cumulative distribution functions (CDF) of this data. Physically, we expect some correlations between these distributions, for example, longer coherence time for qubits should allow for higher fidelity or faster quantum control on the gates and therefore higher single-qubit and two-qubit fidelities.

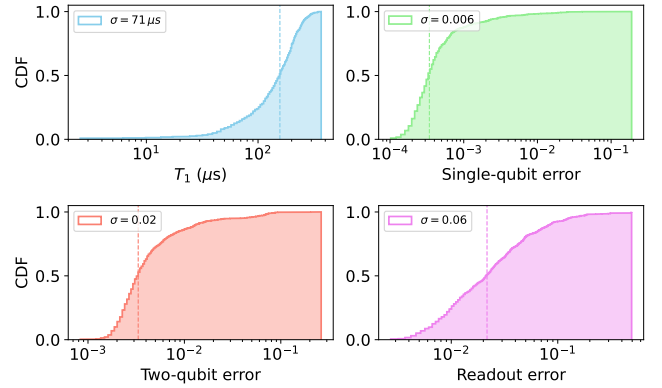


Figure 18: Cumulative distribution functions for T_1 times, and the single-qubit, two-qubit, and readout errors accumulated over nine days for the `ibm_torino` processor [105]. Dashed vertical lines indicate the mean values. Legends indicate standard deviations.

To capture these correlations, we employ a random-forest model [106], which is a common choice of machine learning model for small sets of training data. We use the T_1 time as an input feature, and train three models for the conditional generation of single-qubit, two-qubit, and readout errors, respectively.

The single-qubit and readout errors are predicted from the T_1 time of the corresponding qubit, while the two-qubit error model uses the T_1 time of both qubits as input. Figure 19 shows that our random-forest models adequately estimate the gate and readout errors.

We use these models to study the impact of process variability on logical error rates of QEC codes. This variability is characterized by the standard deviation σ of the distribution. Therefore, we construct several synthetic T_1 distributions with varying values of σ , by rescaling

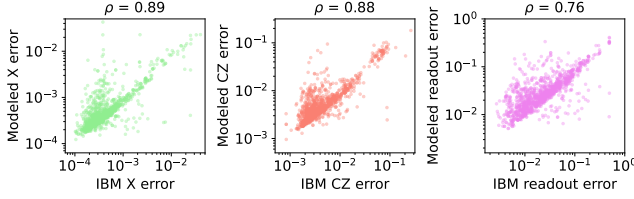


Figure 19: Correlations between the true and predicted single-qubit, two-qubit, and readout errors as a function of the T_1 time. The Pearson correlation coefficients ρ are reported at the top of each figure.

the IBM T_1 distribution,

$$T_1 \rightarrow \mu + a(T_1 - \mu), \quad (1)$$

where μ is the mean of the original distribution and a is the rescaling factor. This transformation ensures that only σ varies while the mean and the higher standardized moments of the distribution remain fixed.

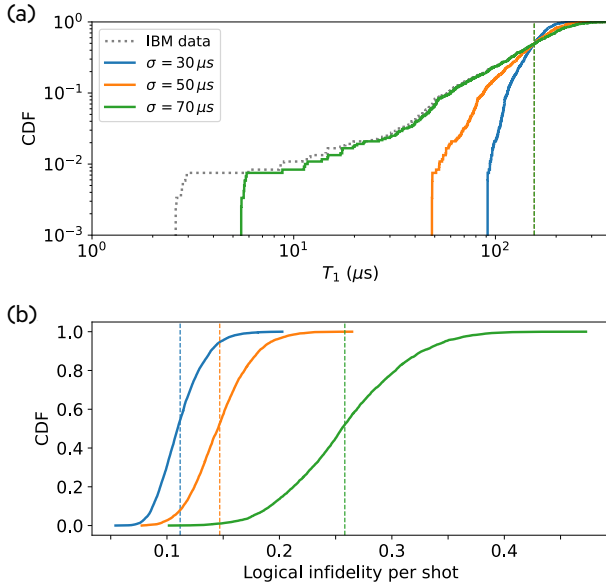


Figure 20: (a) Transformed T_1 distributions with different standard deviation, σ . The dotted grey curve shows the CDF of the original T_1 distribution for comparison. (b) Cumulative distribution functions of the logical infidelities of a rotated surface code of distance $d = 9$. Dashed vertical lines indicate the respective mean values.

In Figure 20(a) we show the CDFs of three distributions generated by applying the transformation (1). The three values of σ are chosen to represent different amounts of reductions in process variability from that of the studied QPU.

We investigate the impact of these distributions on the performance of the logical memory experiment on a distance-9 rotated surface code. To do this, we perform simulations where the gate and measurement errors on each physical qubit on the rotated surface code patch are distinct, better reflecting the experimental reality. For each of the distributions constructed above, we repeat the following process 5000 times:

| Parameter | IBM Values |
|------------------------|------------|
| Single-qubit gate time | 32 ns |
| Two-qubit gate time | 68 ns |
| State preparation time | 780 ns |
| Measurement time | 780 ns |
| Reset time | 780 ns |

Table II: Gate times of the `ibm_torino` QPU.

1. Sample a T_1 time for each physical qubit on the rotated surface code patch.
2. Use the machine learning models to generate synthetic gate and measurement error rates on each physical qubit (and each adjacent pair of qubits in the case of two-qubit gate errors).
3. Simulate the fault-tolerant memory protocol using the assigned gate fidelities, and assuming gate times from the `ibm_torino` data (see Table II), to determine a logical error rate for the code.

The output distributions of the logical error rates are shown in Figure 20(b). We observe that higher values for σ result in a higher logical error rate. This suggests that the impact of a larger number of poor-quality qubits and gates dominates that of the larger number of high-quality qubits and gates. This analysis suggests that QPU manufacturing should not only focus on improving the mean quality of qubits and gates, but also on achieving more-robust fabrication processes so as to avoid heavy tails of poor-quality qubits. Finally, we emphasize that this study is confined to analyzing the impact of the variance of the T_1 distribution. However, it is speculated that the higher moments, such as skewness and kurtosis, might carry valuable signatures for such benchmarking and should be investigated in the future.

E. Emulation of other FTQC protocols

Magic state preparation. A critical process in fault-tolerant quantum computation is preparation of high fidelity logical magic states. These states are produced by first employing a magic state preparation protocol [96–98], which produces relatively low fidelity logical magic states at small distances by employing physical T gates. A large number of such logical states are then grown to higher distances and fed to magic state distillation units to prepare a lower number of higher fidelity magic states. These magic state distillation units are only able to perform if they are fed logical magic states of sufficient fidelity. It has been estimated that the 15-to-1 distillation units have acceptance probability

$$1 - 15P_{\text{magic}} - 356P_{\text{Cliff}}, \quad (2)$$

where P_{magic} and P_{Cliff} are the logical error rates of input logical magic state and Clifford operations respectively

[49]. Hence, we need to produce logical magic states with error probability

$$P_{\text{magic}} < \frac{1 - 356P_{\text{Cliff}}}{15}. \quad (3)$$

Whether magic states of this fidelity can be produced depends both on the specific magic state preparation protocol used and the hardware noise profile. We studied a protocol that cleverly exploits hook-injection errors to create high fidelity relatively low-distance magic states [97] using Clifford simulations. This protocol first uses a physical T -gate to create a magic state on a small rotated surface code patch of distance d_1 . It then post-selects the states for which no errors are detected and grows them to a larger distance d . The simulation results reported in Figure 21 show that the error rates increase with distance d , suggesting that the protocol is not fault tolerant, and explains why distillation units are needed instead of directly growing the magic states to a target distance. For simplicity we substitute the logical Clifford error rate with the logical error rate of the memory protocol in Equation (3) and draw the respective threshold curves for each of the hardware parameters studied in Figure 21. We observe that both target and Λ_{18} parameter sets are significantly below their respective 15-to-1 distillation thresholds, while the baseline set is not.

A more recent protocol [98] demonstrates significant improvements in the logical error rates for magic state preparation. This protocol introduces a number of improvements over past protocols, such as cleverly designed gradual growth stages and appropriate post-selections to ensure that error rates drop when increasing distances in practical regimes of error rates. We use the authors' code to estimate the error rates for our three hardware parameter sets, also shown in Figure 21. We observe that all three hardware parameter sets are significantly below the 15-to-1 distillation threshold for this protocol and therefore the QREs in this paper use this protocol.

Teleportation of logical qubits. To perform lattice surgery fault tolerantly, both space-like and time-like errors must be corrected. As discussed and numerically demonstrated in [107], space-like errors are exponentially suppressed by increasing the code distance of the logical qubit. Similarly, time-like errors are exponentially reduced by increasing the number of stabilizer measurement rounds during the merge operation (see Figure 22, middle). In this context, the number of parity check cycles conducted while the two logical patches are merged is referred to as the temporal code distance. To evaluate the overall success rate of lattice surgery, we assess logical qubit teleportation under varying space-time parameters.

To determine the success rate of logical qubit teleportation, we estimate the average state infidelity defined by $P_a = \frac{1}{6} \sum_{\psi} P_{\psi}$, where P_{ψ} represents the infidelity of teleporting each logical state $|\psi\rangle \in \{|0_L\rangle, |1_L\rangle, |+_L\rangle, |-_L\rangle, |+i_L\rangle, |-i_L\rangle\}$. Note that, the process fidelity is then given by $F_p = \frac{(D+1)(1-P_a)-1}{D} = 1 - 3/2P_a$, where $D = 2$ is

the dimension of the Hilbert space of the single qubit being teleported [108, 109]. Since accessing the Y operator of the surface code is cumbersome [110], we teleport the logical states $|\psi_L\rangle \in \{|0_L\rangle, |+_L\rangle\}$ using an XX (rough) merge, as illustrated in Figure 22. From these simulations, we approximate the average state infidelity as $P_a \approx \frac{2}{3}(P_+ + P_0)$, providing an overestimate of the infidelity. The teleportation protocol we study is depicted in Figures 22 and 23 and outlined as follows:

1. Preparation: We begin by perfectly preparing the logical source state $|\psi_L\rangle \in \{|0_L\rangle, |+_L\rangle\}$ and the target state $|0_L\rangle$. These states are stabilized for r_{pm} rounds, as shown in Figure 22 (left).
2. Merging: After the round r_{pm} , the bus data qubits are initialized in the physical $|0\rangle$ state and the rough edges of the two surfaces are merged. Then the entire surface is stabilized for r_m rounds to determine (a possibly erroneous) measurement of the $X \otimes X$ observable with outcome m_1 , as shown in Figure 22 (middle).
3. Splitting: After the round $r_{pm} + r_m$, the bus data qubits are measured in the Z basis (splitting) and the remaining patches are stabilized for r_s rounds.

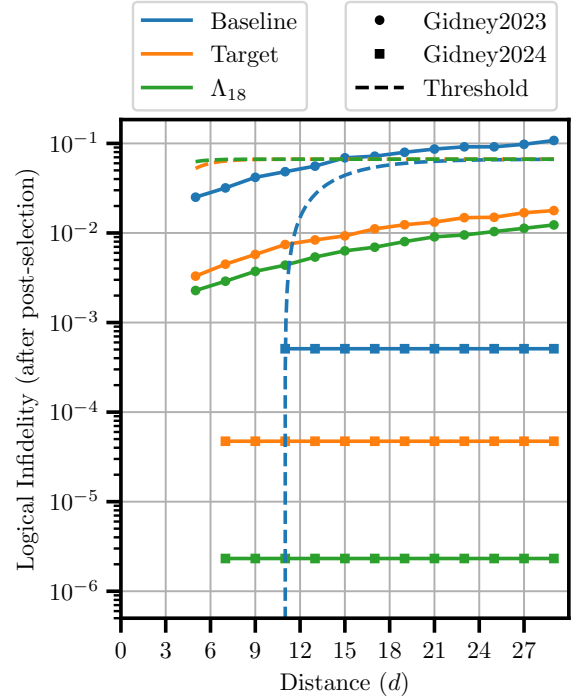


Figure 21: Performance of the magic state preparation protocols Gidney2023 [97] and Gidney2024 [98] for the three parameter sets. Here, we fix $d_1 = 3$, because it yields the best performance for the baseline parameter set. The dashed curves indicate the distillation unit input threshold (3) for the baseline, target, and Λ_{18} parameter set, respectively, where the difference is due to the fact that P_{Cliff} , estimated from the memory protocol logical error rate, depends on hardware noise. Observe that the baseline set is below the respective 15-to-1 distillation threshold only for the Gidney2024 protocol, while it is above threshold for Gidney2023.

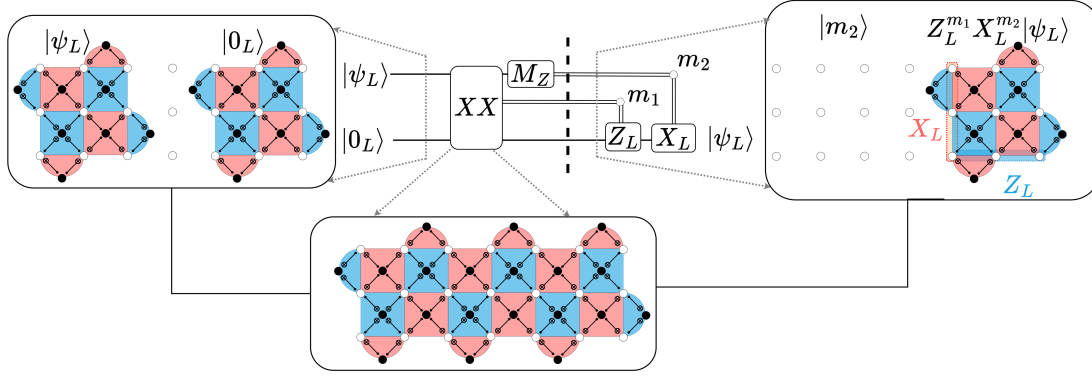


Figure 22: Logical teleportation via lattice surgery for a rotated surface code of distance $d = 3$ and bus width $b = 1$. Illustrated are the states before merging (left), during merging (middle), and after splitting (right). The wire diagram of the teleporation quantum circuit including the relevant classical feedback and the needed recovery operations is also shown. We simulate the portion of the circuit before the dashed line using a circuit-level noise model. Note that at the dashed line invoking a decoder may flip the values m_1 and m_2 . We do not simulate the circuitry of the recovery operations Z_L and X_L ; instead, we measure all qubits at the dashed line to determine the performance of the teleporation.

Note that this also results in a perfect round of syndrome measurements on the bus patches.

4. Projection: After the round $r_{pm} + r_m + r_s$, the source data qubits are measured in the Z -basis, to determine the (possibly erroneous) value of m_2 . The projection of the source data qubits is illustrated in Figure 22 (right).
5. Recovery: At this point, by invoking a decoder, the values of m_1 and m_2 may be corrected and the recovery operations Z_L and X_L are conditionally applied.

To estimate P_a at large distances we simulate the above-mentioned teleportation protocol for varying spatial code distances d , $3 \leq d \leq 15$, and temporal code distances r_m , $5 \leq r_m \leq d$, fixing the bus width $b = 3d$ and incorporating the circuit-level noise model detailed in Section III B. We regress the following predictive models from the obtained numerical results of P_0 and P_+ values to predict the fidelity of the protocol at high distances [50, 107]:

$$P_0 = \mu_X (2d + b) r_m \Lambda_X^{-(d+1)/2}, \quad (4)$$

$$P_+ = \mu_Z d \Lambda_Z^{-(d+1)/2} + \mu_T d b \Lambda_T^{-(r_m+1)/2}. \quad (5)$$

In our simulations the pre-merge stabilization rounds is fixed to $r_{pm} = 1$, during which the two logical states are prepared (perfectly). We also use $r_s = 0$ as illustrated Figure 23(a). For our benchmark purposes the recovery step is not performed, instead, the target data qubits are also measured in the basis corresponding to the initial source state $|\psi_L\rangle$ to determine the teleported logical state on the target.

In Figure 24(a), we report the logical error rates of teleporting the states $|+_L\rangle$ and $|0_L\rangle$ (labeled P_+ and P_0 , respectively) as a function of the code distance for $3 \leq d \leq 15$ with corresponding bus width $b = 3d$ and temporal code distance $r_m \in \{d, 3d\}$. We highlight two observations from Figure 24(a).

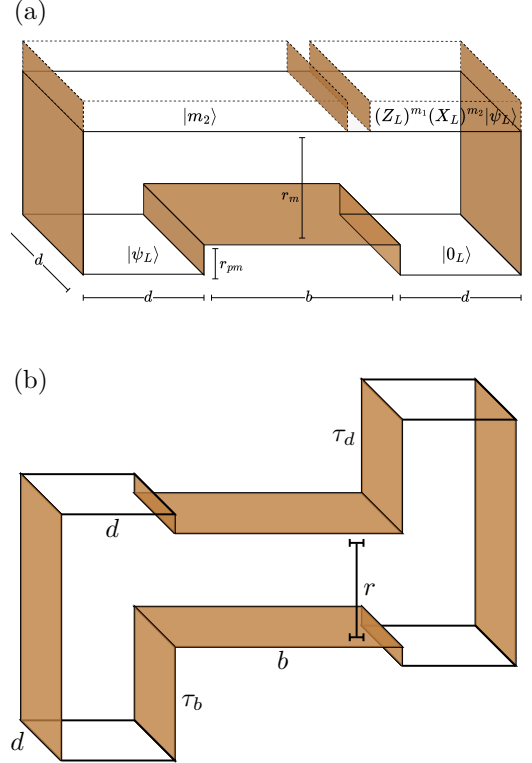


Figure 23: **space-time diagrams for teleportation via lattice surgery.** (a) space-time diagram of teleportation where $r_s = 0$ QEC cycles are performed after splitting. The spatial dimensions are the same as shown in Figure 27, and the temporal dimension is divided into r_{pm} pre-merge rounds and r_m merge rounds. (b) The effect of buffer and decoder delays on lattice surgery fidelities. For each logical operation in the core processor, a teleportation is implemented involving a magic state (the left code patch) and data qubits (represented by the right code patch). The magic state may incur a delay τ_b in the buffer before the surgery is performed. The source (magic) state and the bus are measured out after the merge operation but target patches must await the decoder decisions, available after a decoder delay time, τ_d . These idling patches must be protected using further stabilization rounds (quantum memory) during this period; therefore, accumulation of further errors is inevitable and must be taken into account by the compiler, FTQC emulator, and resource estimators.

- The error rates are suppressed as a function of code distance for both states. This indicates that the noise parameters are below threshold.
- Increasing r_m decreases the teleportation fidelity of $|0_L\rangle$ while increasing the fidelity of teleporting $|+_L\rangle$ (see also Figure 24(d)).

In Figure 24(b), we show the fitting for the X and Z -type terms of Equations (4) and (5). This model predicts the X and Z -type errors better at the high- r_m regime (hence choosing the $r_m = 3d$ data). Similarly, in Figure 24(c) we estimate the time-like error suppression term in Equation (5) at the high-distance regime, obtaining $\mu_T \approx 0.0273(6)$ and $\Lambda_T \approx 1.967(6)$. This information is sufficient to estimate the average error rate P_a of high-distance teleportations.

As another application, in Figure 24(d), we find the optimal number merge stabilization rounds r_m for a given distance d . Here we choose $d = 7$ and use two sizes for the bus $b \in \{d, 3d\}$. We plot the estimated teleportation fidelity $P_a \approx \frac{2}{3}(P_0 + P_+)$ as a function of r_m and note that the minimum of each curve is at $r_m > 7$, highlighting the fact that the optimal number of QEC rounds for a lattice surgery operation with code distance d may deviate from the commonly assumed value, d .

For teleportation of magic states in our core processor, the source (magic) state has resided in the buffer for some average expected buffer delay time, τ_b (which can be as low as 1 clock cycle for balanced production and consumption of the magic states). The targets of teleportation are logical data qubit patches in the core processor for which further QEC rounds are executed until the decoder outcome is available. We denote this delay by τ_d . Inclusion of the buffer and decoder delays and assuming an average rate for all types of surgeries results in the model

$$\mu[d(2r + \tau_b + \tau_d + 1) + br]\Lambda^{-(d+1)/2} + \mu_T db \Lambda_T^{-(r+1)/2}, \quad (6)$$

which still distinguishes time-like and space-like errors but ignores the type of surgery; e.g., XX merge or otherwise (Figure 23(b)).

F. High-performance real-time decoding platform

Challenges and requirements. The high speed of superconducting processors, a great advantage for utility-scale applications, requires well-engineered control and decoder architecture, both on the software and hardware levels. A key technical challenge is that decoding simultaneously requires peta-scale computation and low latency for real-time feed-forward. Furthermore at stage, it isn't known yet what algorithms are most effective at decoding, therefore there is a systems engineering trade-off between on performance and flexibility.

Performing universal fault-tolerant quantum computation using QEC mandates feedforward-based implementation of certain quantum gates (e.g. T gates) with low

latency [111]. In these implementations, a conditional operation is applied based on the result of a logical measurement as well as the decoding of syndromes of many previous QEC cycles. The classical feed-forward latency is measured from the physical execution of the logical measurement until the controller executes a conditional gate (L0 and L1 in Figure 25(c)).

For efficient execution of fault-tolerant feed-forward gates, the decoder needs to be ready in time for the conditional gate execution. We note that on average, d (distance) cycles are allowed for the decoder result for multiple reasons. First, when the conditional gate is followed by gates that commute, it may be deferred after the gates that do not depend on the decoding result. Second, the gates that follow the conditional gate may require synchronization with other surfaces, allowing to defer the conditional gate without impacting the circuit. In addition, we note that sporadic delays caused by the decoder have a small impact on the overall performance as long as on average the decoder result is ready on time, as shown in Figure 25(b). Therefore we target an end-to-end average decoding latency shorter than d QEC cycles, which implies a target latency of approximately 10 μ s. To meet these latency targets and implement QEC decoding efficiently, it is essential to optimize the performance of both the control-decoder communication channel and decoding task. For the controller-decoder channel, throughput must exceed the data generation rate. The controller should locally perform state discrimination including optional soft readout indicators, encoding each qubit state with a minimal bit representation. With 20K qubits, a 4-bit state representation per qubit, and a QEC cycle time of 550 ns, a minimum net bandwidth of 150 Gb/s is required. Additionally, data sent from the decoder to the controller must be efficiently compacted to communicate only the necessary logical instruction. In addition, the overall latency should be minimal, including readout state discrimination, data aggregation from multiple controllers and transmission to and from the decoder.

The decoding processes, which include multiple concurrent decoding processes, should be designed to minimize latency. Scalable hardware is required, as decoding for circuits with 10K–100K qubits demands extensive computational capacity. Some decoding algorithms, such as Fusion Blossom, may exhibit variability in the decoding time dependent on the error pattern. The QEC implementation should be designed to accommodate this variability. For the decoder not to limit the performance, the average throughput of the decoder should exceed the syndrome generation rate. In addition, the decoder average latency including the roundtrip communication time should be shorter than the d QEC cycles.

Real-time decoding architecture. To address the FTQC requirements, a proposed architecture, illustrated in 12 is designed to allow a high-performance decoding platform along with low latency communication between the quantum control clusters and the decoding platform. The control clusters control one or more quantum sur-

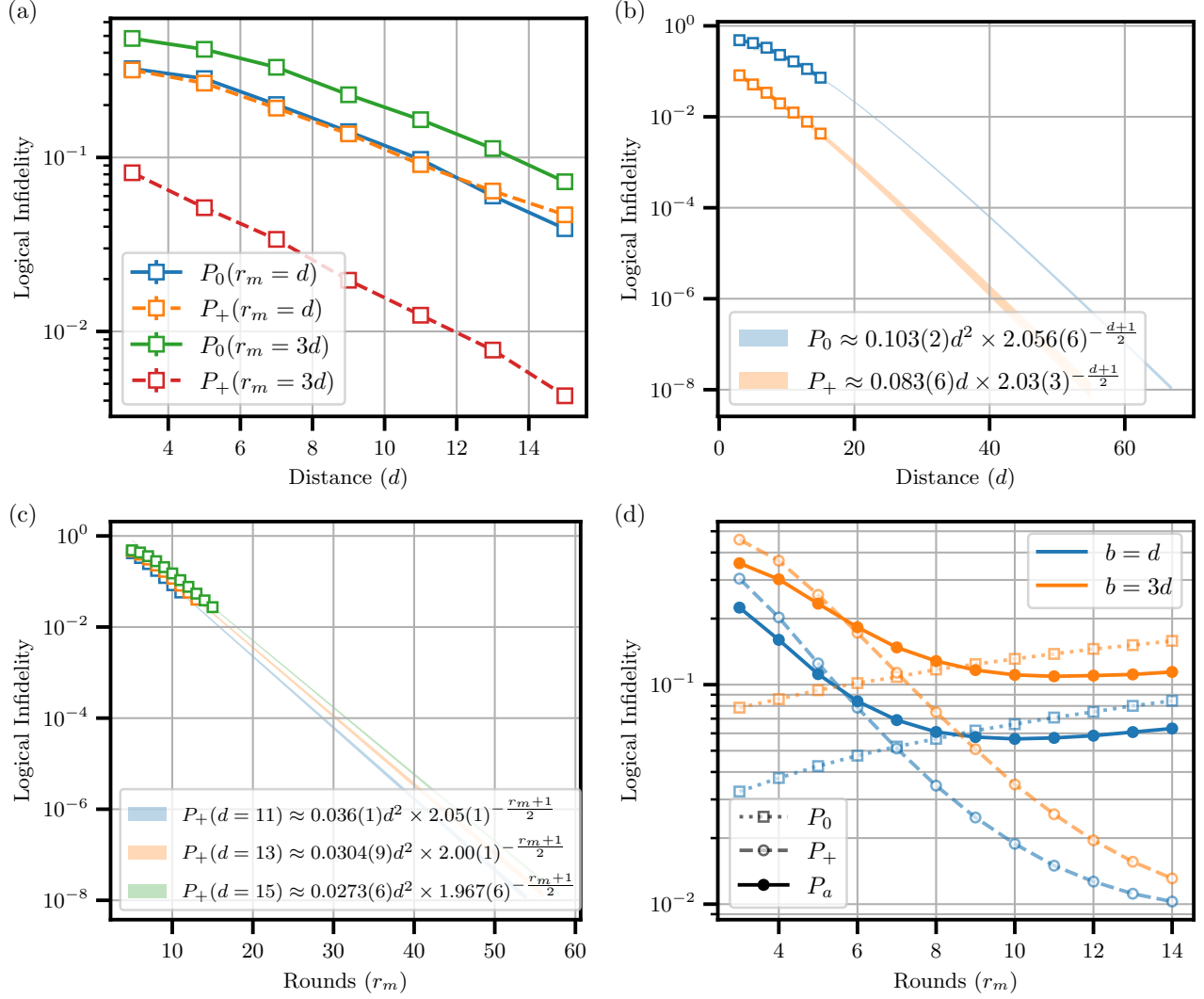


Figure 24: (a) Error suppression as a function of code distance d for bus width $b = 3d$, merged stabilization round $r_m \in \{d, 3d\}$ for teleportation of $|0_L\rangle$ (P_0) and $|+L\rangle$ (P_+). (b) An exponential fit for $P_0(r_m = 3d)$ and $P_+(r_m = 3d)$ using an exponential functions of the form $P_0 \approx \mu_X d^2 \Lambda_X^{-(d+1)/2}$ and $P_+ \approx \mu_Z d \Lambda_Z^{-(d+1)/2}$. (c) An exponential suppression of P_+ as a function of the number of rounds for rounds $r_m < d$, for which we expect the logical error suppression of the form $P_+ \approx \mu_T d^2 \Lambda_T^{-(r_m+1)/2}$. (d) Logical teleportation error rates for teleporting the states $|+L\rangle$ and $|0_L\rangle$ as a function of the merged stabilization rounds (temporal code distance) for the code of distance $d = 7$ over a bus of width $b \in \{d, 3d\}$ and the corresponding average state infidelities calculated using $\frac{2}{3}(P_0 + P_+)$.

faces, operating in synchronization to execute QEC cycles based on the state of each surface. In addition, the control clusters are responsible to benchmark the qubits performance and maintain qubits calibration. The acceleration servers provide a high-performance platform for the execution of decoder instances and the logical circuit orchestration. They are based on CPU/GPU processors with direct data transfer capabilities to and from the control system. We note that GPUs are beneficial for real-time decoding of QECCs thanks to their massive parallelism and their high-bandwidth/low-latency interfaces. In addition, the servers may incorporate specialized ASIC or FPGA acceleration cards and dedicated hardware offload capabilities. In large-scale systems, multiple acceleration servers may operate in parallel, running multi-

ple decoder instances. The control clusters and accelerator servers are connected by a low-latency network. The network facilitates the aggregation of readout data from the control clusters and distribution of the data to the appropriate decoders. To meet the end-to-end latency requirements, the total time for data aggregation, roundtrip communication and decoding should be in the order of 10 us, which require the design of a specialized communication protocol for QEC.

Real-time decoding on a DGX Quantum platform. DGX Quantum provides a tight integration of QM's OPX1000 controller with Nvidia's Grace Hopper (GH) superchip and offers an effective platform for FTQC, supporting both logical circuit orchestration and decoding processes. The close integration of CPU and

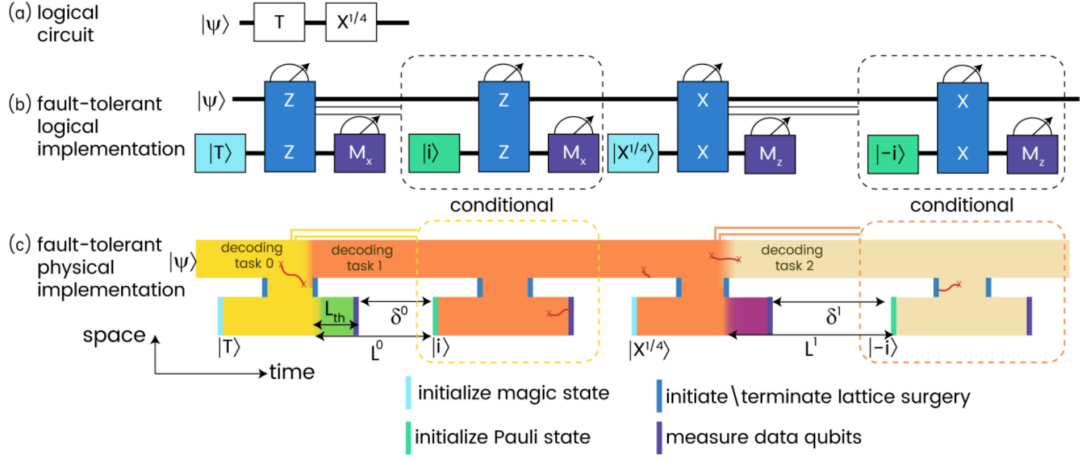


Figure 25: Example of non-Clifford computation with surface codes. (a) An example of a logical circuit containing two non-Clifford gates. (b) The fault-tolerant logical circuit that implements the circuit in (a) with surface codes with only a single ancillary surface. The dashed square denotes the feed-forward conditional logical gates that verify that the planned circuit is executed. (c) The space-time view of the circuit in (b) with surface codes. Each colour denotes a separate decoding task, chosen to end each task with a logical measurement. The decoding outcome of the lattice surgery between a magic state surface and the computation surface determines a feed-forward circuit, which delays the circuit by if the feed-forward latency (L) is larger than a threshold latency (L_{th}).

GPU resources enables real-time, parallel execution of various decoding algorithms, including deep-learning-based distributed decoders. The system is connected to the low-latency QEC network and transfers data from the control system to the high-performance CPU-GPU and vice-versa over a PCIe interface. A round-trip feed-forward latency (not including the decoding task) from measurement to the decoder and back to conditional gate control, has been benchmarked at less than 3.8 μ s.

The DGX Quantum platform is designed to be connected to a hierarchical, scalable QEC network for data aggregation and distribution, ensuring low latency across systems with 10K qubits and beyond. The standard, modern PCIe interface provides 100s of Gb/s of bandwidth and supports also communication with connected ASIC or FPGA acceleration cards. As scaling extends to systems with 10K–100K qubits, a dedicated, optimized interface connecting the QEC network directly to the decoders may be desired, to further reduce the latency.

DGX Quantum leverages the extensive parallel processing capabilities, large memory, and high memory bandwidth of the GPU, providing a robust platform for QEC decoding and runtime execution. Moreover, as a software based solution, DGX Quantum provides a platform for flexibility and rapid development that are desired in the early stages of FTQC.

Preliminary evaluation of a software-based implementation of Fusion Blossom algorithm with batch decoding on a DGX Quantum server demonstrated that a serial implementation could sustain the necessary decoding throughput for a distance $d=11$, with a basic error model with error probability $P_{error} = 110^{-3}$. In the next steps, we plan to implement Fusion Blossom in stream mode, in addition to leveraging parallel processing and utilizing the large memory capacity for potential caching and

optimization for common error patterns.

End-to-end testing of the QEC system, including control, decoders, and runtime, is desirable prior for qubits availability at this scale. The DGX Quantum system can emulate a larger-scale setup by generating synthetic syndromes based on a given error model and loading them to the control system. To minimize an impact on the server under test, a separate server could be dedicated to the emulation, leveraging the system's support for multiple server instances. In this setup, the control system streams syndrome data to the QEC server under test using the low-latency communication interface, which then updates the control state. This emulation enables measurement of end-to-end latency, providing a comprehensive engineering perspective on system bottlenecks and opportunities for architectural optimization.

G. Distributed FTQC across multiple dilution refrigerators

A fault-tolerant quantum computer with 1M+ physical qubits may require multiple dilution refrigerators (DR) with quantum interconnects between DRs. The inter-DR and intra-DR architecture of the computer is discussed further in Section III G. The assembler prioritizes inter-DR lattice surgeries (involving less than 120k qubits) over multi-fridge surgeries, as logical teleportation of states between different DRs is much slower and of lower fidelity than intra-DR operations. Multi-DR surgeries involving code distance d will require at least d optical interconnects between nearest-neighbour DRs, which is a demanding requirement [44–48].

Assembling large FTQC programs among multiple DRs. Embedding FTQC architectures across mul-

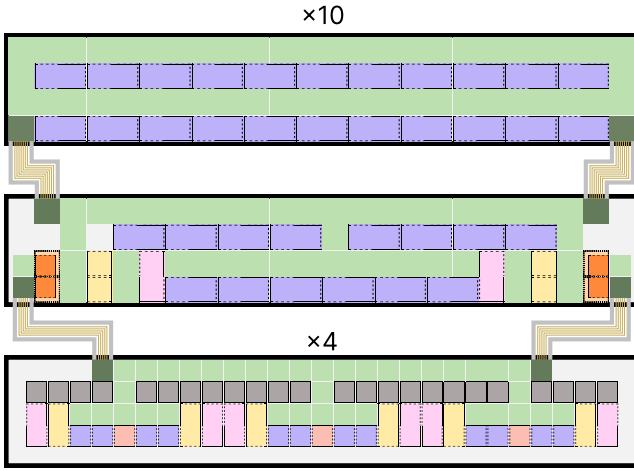


Figure 26: Example of an embedded multi-DR architecture for executing a quantum circuit associated with electronic-structure quantum simulation based on qubitization of the *p*-benzynes molecule for an active space involving six molecular orbitals (see Section IV). Our estimates indicate that this circuit requires at least 11 distillation units in the MSF with a code distance of 21 to ensure a continuous supply of magic states to the core processor, which requires 341 data qubits with a code distance of 25. The architecture shown consists of 15 DRs, each containing 120,000 physical qubits. Four DRs are configured with three distillation units each, 10 DRs accommodate 33 data qubits each, and the remaining DR includes additional data qubits along with dedicated zones for magic state growth and storage. Multi-qubit lattice surgery is performed using the quantum bus, while magic states are transferred between DRs using as many optical interconnects as the code distances involved.

multiple DRs involves solving a complex embedding problem to determine the connectivities between DRs, teleportation sites adjacent to the interconnects, and appropriate areas across the multi-DR system for the core processor and the MSF zones of required code distances. The embedding prioritizes intra-DR connectivity to ensure the robustness of FTQC protocols against noise introduced by the imperfect interconnects. This is also an important consideration even within individual DRs when lattice surgeries span across the edge couplers of the QPU (i.e., the weaker capacitive coupling between the 20k qubit wafers).

Designing the layout of the core processor and the MSFs, including the shapes of the distillation units, is critical for fitting them within the available space. Figure 26 illustrates an example of an embedded multi-DR architecture designed for executing the *p*-benzynes circuit generated using qubitization with an active space of 2 based on the data generated in Section III C.

Impact of noisy optical interconnects. To study the impact of both types of weak couplers described above, we have rerun the teleportation experiments of Section III E by incorporating columns of weaker CNOT gates (called “cuts”) between the surface code patches as illustrated in Figure 27. In Figs. 28(a) and (b), we show that using weaker CNOTs of infidelity $p_{\text{link}} = 0.01$ and fixing all other coupler infidelities to the baseline value $p_{CX} = 0.003$ does not significantly affect the teleportation fidelity even for 4 cuts within the

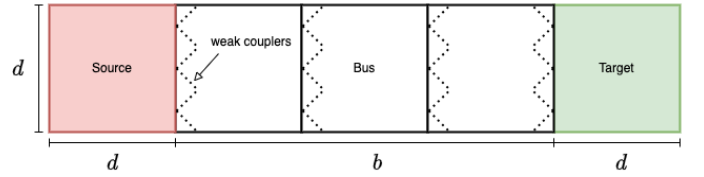


Figure 27: A time slice of the logical teleportation protocol via lattice surgery in a multi-DR distributed architecture where the cuts (weak couplers) are illustrated in broken lines and placed regularly along a bus of dimension $b \times d$.

bus. However, much weaker interconnects can be problematic as shown in Figs. 28(c) and (d) where we observe a thresholding behavior at about $p_{\text{link}} \approx 0.06$.

We conclude that distributed surface code architectures across multiple dilution refrigerators can tolerate two-qubit errors on the order of 1% arising from noisy optical interconnects between the DRs. However, the logical performance rapidly deteriorates as these errors become significantly worse, and for error rates beyond 5%, achieving fault tolerance becomes problematic.

IV. RESOURCE ESTIMATION FOR FAULT-TOLERANT QUANTUM COMPUTATION

To evaluate how attaining an improved quality of physical qubits and gates affects the overall resource requirements of FTQC, it is crucial to conduct detailed physical quantum resource estimations (QRE) for practical applications at utility scale. Our numerical QREs aim to compare the concrete FTQC resource requirements for the three sets of hardware parameter specifications summarized in Table I. These QRE studies were conducted using automated tools of TopQAD [50, 51, 99] and AzureQRE [112].

A. Quantum computation of electronic spectra as a representative high-utility application

One of the most promising applications of quantum computing is solving quantum chemistry problems. An important representative computational task in quantum chemistry is estimating ground-state energies of molecules. While this task is classically tractable for small molecules using advanced classical algorithms developed in the field of traditional quantum chemistry [113], electronic-structure simulations of large molecules are widely considered to be intractable for classical computers. Here, for the purpose of demonstrating the practicality of solving such problems on a quantum computer, we present QRE studies of electronic-structure quantum simulations for two molecules of high practical interest. The first molecule analyzed is the biradical *para*-benzynes molecule (*p*-benzynes), which has the molecular formula C_6H_4 . Its energetically lowest configuration

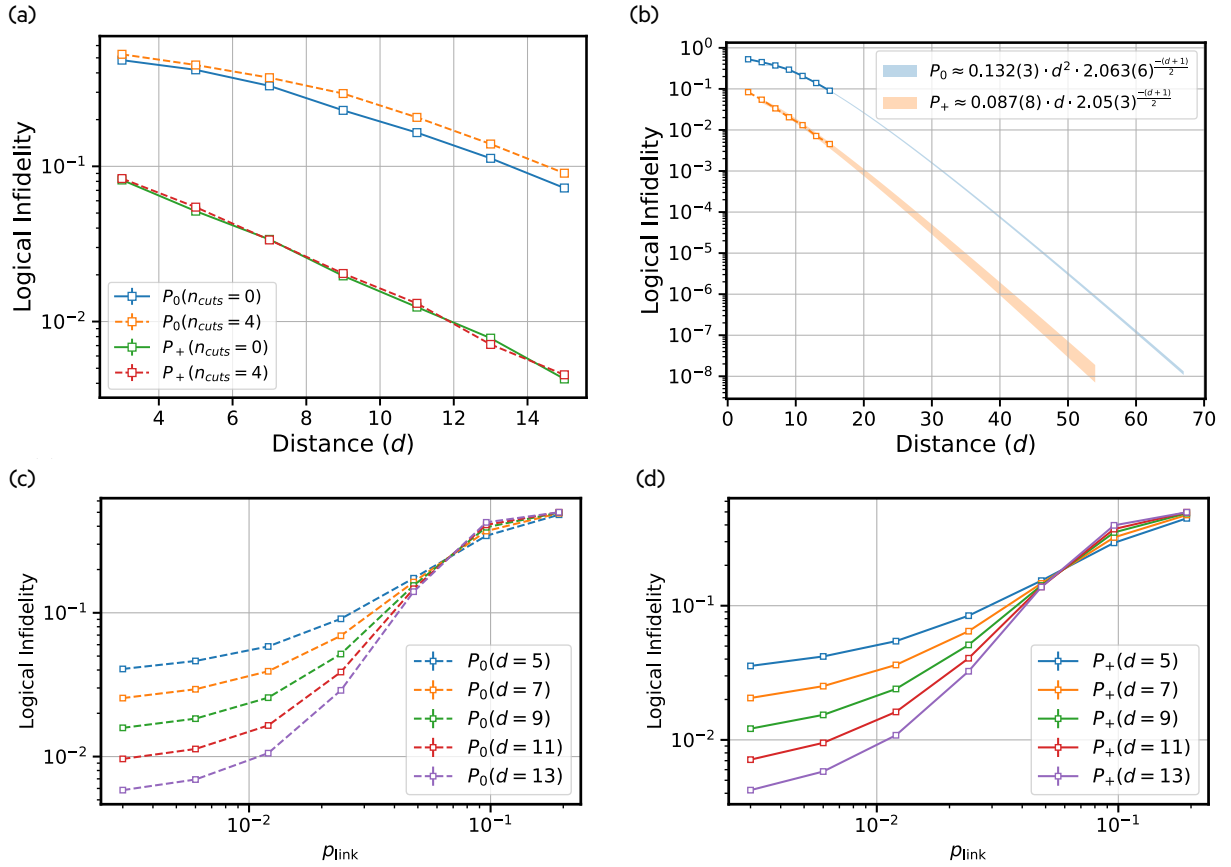


Figure 28: (a) Logical infidelity as a function of code distance for teleportation of $|+_L\rangle$ and $|0_L\rangle$ states with and without cuts for corresponding values of bus width $b = 3d$ and temporal code distance $r_m = 3d$. (b) Exponential fits for P_0 and P_+ for $n_{cuts} = 4$. The obtained μ and Λ values are close to that of Figure 24(b). Logical infidelity of teleporting $|0_L\rangle$ (c) and $|+_L\rangle$ states (d) as a function of the infidelity of the weak CX gates in the cuts.

is formed by a singlet biradical. Among other applications, its reactivity has the potential to play an important role in the design of enediyne drugs with high antitumour or anticancer properties [114, 115]. The second molecule analyzed is the iron-molybdenum cofactor (FeMoco) of nitrogenase, which acts as a crucial catalyst in the process of biological nitrogen fixation. This molecule, as well as many others, have been used as representative targets for future quantum simulators in several recent works [35, 36, 116–118].

For our QRE studies, we first generate the logical quantum circuits associated with electronic-structure simulations of the p -benzynes and FeMoco molecules. More concretely, we analyze the resource requirements associated with estimating the energy of the ground states of these molecules using the well-established quantum phase estimation (QPE) algorithm [119, 120]. For the p -benzynes molecule, we analyze the singlet ground state using a variety of active spaces that are specified below; for the FeMoco molecule, we analyze the active-space model proposed in Ref. [36].

Our studies rely on electronic-structure simulations in the second quantization framework of quantum theory. Numerous software tools exist to derive the second-

quantized Hamiltonian from a molecule’s specifications that fully characterize the quantum system. Basic molecular specifications include the types of atoms that constitute the molecule and the molecule’s geometry (typically summarized in an `xyz` file), total charge, and the total spin. In addition, a basis set $\{\phi_\alpha(\mathbf{x})\}$ must be selected to represent the fermionic orbitals, which in the language of second quantization are occupied or unoccupied, represented by occupation number states and fermionic creation and annihilation operators (\hat{a}_α and \hat{a}_α^\dagger) acting upon them. Furthermore, to reduce the computational cost, a common approach is to restrict computations to a reasonably chosen active space involving only a subset of the chosen orbital basis set. Finally, the model Hamiltonian associated with an active space is translated from the second quantization framework to a framework suitable for the quantum circuit model. This fermion-to-qubit mapping is typically accomplished via either the Jordan–Wigner [121] or the Bravyi–Kitaev [122] transformations. To derive the model Hamiltonians for the various active spaces associated with p -benzynes, we used Tangelo [123], an open source Python software package for end-to-end chemistry workflows. For the FeMoco molecule, we used the FCIDUMP file provided as part of

the data and code repository [124] of Ref. [117].

The standard QPE algorithm [120] samples in the eigenbasis of the molecular Hamiltonian H by measuring the phase that is accumulated on an initial input quantum state through multiple controlled executions of the time-evolution operator $\exp(-iHt)$. Its most resource-intensive part consists in implementing the unitary $\exp(-iHt)$ by a quantum circuit, along with repeating this circuit a number of times that scales as $\mathcal{O}(1/\epsilon)$ for an allowable target error ϵ in phase estimation. An alternative approach to sampling the spectrum of the molecular Hamiltonian H via phase estimation is based on the framework of qubitization [125]. Indeed, most of the recent QRE studies on electronic-structure quantum simulations rely on the qubitization framework (see, e.g., [49, 117, 126–132]). This approach uses a new operation called *qubiterate* that is akin to the quantum walk operator $e^{i \arccos(H/\lambda)}$ (where λ is typically the sum of the absolute values of the weightings in the molecular Hamiltonian). Since the qubiterate’s eigenvalue spectrum can be obtained from that of the unitary $\exp(-iHt)$ via an arccos transformation, the former can be used in QPE in place of the latter, as proposed in Refs. [133, 134]. An advantage of this approach is that steps of a quantum walk can be implemented exactly, assuming access to arbitrary single-qubit rotations, in contrast to all approaches that are based on Hamiltonian simulation of the time-evolution operator $\exp(-iHt)$, which can only be approximated.

Moreover, aiming to reduce algorithmic complexity, the majority of recent literature on electronic-structure quantum simulations and the associated resource requirements has focused on combining the technique of qubitization applied to molecular systems with various tensor factorization techniques for the Coulomb operator. State-of-the-art algorithms of this type include the single low-rank factorization algorithm of Berry et al. [126], the double low-rank factorization algorithm of von Burg et al. [127], and the tensor hypercontraction algorithm of Lee et al. [117], resulting in a continual improvement of the T gate or Toffoli gate complexity from $\mathcal{O}(N^5/\epsilon^{3/2})$ for the Trotter-based approach to $\mathcal{O}(N\lambda/\epsilon)$, where λ is the 1-norm of Hamiltonian coefficients which typically has a scaling between $\mathcal{O}(N)$ and $\mathcal{O}(N^3)$.

B. Quantum resource estimation studies for *p*-benzyne and FeMoco

In this section, we present QRE studies for two algorithmic approaches: the Trotter-based approach, and the qubitization-based double low-rank factorization algorithm originally proposed in Ref. [127] and further analyzed in Ref. [117]. Moreover, for the Trotter-based algorithm, we report QRE results for two methods to ensure quantum computations within a target precision (for a discussion in greater detail, see Appendix A 2): the first approach is based on using rigorous analytic bounds on

the errors resulting from the use of Trotter–Suzuki approximation and Trotterization, thus yielding a worst-case number of Trotter slices; the second approach relies on more-realistic Trotter numbers obtained through extrapolation from empirical studies of the Trotter–Suzuki errors for small circuits. In Appendices A 1 to A 3, we elaborate on the workflow for generating the associated logical circuits and analyze the various bounds on the errors incurred in this process, as well as how we choose these bounds to ensure that quantum simulations achieve a given target accuracy.

We report estimates for concrete physical resources required for a fault-tolerant implementation of the QPE algorithm for the *p*-benzyne and FeMoco molecules based on either the second-order Trotter–Suzuki formula used to approximate $\exp(-iHt)$ for the molecular Hamiltonian or the double-factorized (DF) qubitization algorithm of von Burg et al. [127]. In both cases, we assume access to a quantum state with significant overlap with the ground state as input to QPE, for example, a Hartree–Fock state. We do not include the cost of preparing this initial state in our resource estimations. It is worth emphasizing, however, that QPE is only provably fast for problems when the initial state is an eigenstate. When it is not an eigenstate of the Hamiltonian, there is a sampling overhead because the initial state is a superposition of eigenstates. This is a very problem-dependent challenge, but it can be ameliorated by classical preprocessing, that is, by running calculations on a classical computer to generate a better initial state which is then loaded into the quantum computer with a short quantum circuit. For example, one recent work [75] presents an estimate that by using a matrix product state of bond dimension 4000, an overlap of 0.96 can be obtained for the ground state of the FeMoco molecule by implementing a circuit composed of nearly 10^9 Toffoli gate.

The overall cost (in terms of, e.g., T -gate or Toffoli-gate count) of implementing the QPE algorithm can be bounded by (see Ref. [64])

$$\mathcal{O}\left(\frac{g(\epsilon_{\text{QPE}})\Omega}{\epsilon_{\text{QPE}}}\|W'(H)\|^{-1}\right). \quad (7)$$

Here, ϵ_{QPE} is the desired error tolerance in phase estimation; $W(H)$ represents the unitary operator (as a function of the Hamiltonian H) used in the QPE algorithm (e.g., $W(H) = \exp(-iH\tau)$ in the case of Hamiltonian simulation for some time τ , or $W(H) = e^{i \arccos(H/\lambda)}$ in the case of qubitization); Ω denotes the cost of a primitive circuit used to realize the implementation of $W(H)$ (such as a Trotter step in the Trotterization approach, or the LCU oracles associated with qubitization); and $g(\epsilon_{\text{QPE}})$ denotes the number of times that the primitive circuit must be repeated to ensure that the error in the spectrum of H resulting from phase estimation of the eigenphases of $W(H)$ is at most ϵ_{QPE} . Note that, in the qubitization approach, the operator $W(H) = e^{i \arccos(H/\lambda)}$ can typically be implemented as a quantum circuit exactly without approximations beyond those required for the

synthesis of arbitrary-angle rotation gates; this implies $g(\epsilon_{\text{QPE}}) = \mathcal{O}(1)$. Hence, due to $\|W'(H)\|^{-1} \leq \lambda$, the overall cost of QPE in qubitization-based approaches becomes $\mathcal{O}(\Omega\lambda/\epsilon_{\text{QPE}})$. Various versions of QPE have been analyzed in the literature, aiming to reduce its cost. For example, the standard QPE algorithm [120] allows the estimation of eigenvalues within a target error ϵ_{QPE} with probability at least $1/2$ using $\lceil 16\pi/\epsilon_{\text{QPE}} \rceil$ applications of the unitary $\exp(-iHt)$. However, more optimized QPE strategies can achieve the multiplying factor (see [35, 64, 117])

$$M := \lceil \pi/(2\epsilon_{\text{QPE}}) \rceil \quad (8)$$

The QRE analyses presented in this section are based on using this repetition factor. For example, the gate cost of the qubitization-based QPE algorithm is computed as $M\lambda\Omega$, where Ω comprises the costs of the LCU oracles SELECT and PREPARE.

In Tables III and IV, we summarize our logical and physical QRE results for a number of circuits specified by the active space with sizes characterized by the number of orbitals N_{orb} , the number of logical qubits involved in the computation, and the overall allowable target error for QPE. To achieve chemical significance, the overall target error in ground-state energy estimation should be at least within “chemical accuracy”, that is, $\epsilon \leq 1.6$ mHa (see, e.g., Ref. [135]). Energy estimations within chemical accuracy are often sufficient to predict important chemical properties such as chemical reaction rates, but even higher accuracies may be required for quantitatively precise predictions. Here we report resource estimates for two different precisions specified by either the allowable target error $\epsilon = 1.0$ mHa (for qualitative accuracy) or the much lower target error $\epsilon = 0.1$ mHa (for quantitative accuracy), using a circuit-level error budget of 0.01, respectively. The chemical basis set 6-31G is used to represent the spin orbitals in the case of *p*-benzynes, while for the FeMoco molecule we use the active-space model proposed by Li et al. [36].

In the case of the Trotter-based approach, physical runtimes for a complete implementation of QPE are obtained by multiplying the physical runtime for a single Trotter slice by the number of Trotter slices, and then by the number of controlled applications in QPE given by the value of M in Equation (8). For the overall error budget of $\epsilon = 0.1$ mHa, we obtain a value of $\epsilon_{\text{QPE}} = 0.065$ mHa as an optimal choice (see appendix A 2), yielding $M = 24,167$; for the overall target error $\epsilon = 1.0$ mHa, we use $\epsilon_{\text{QPE}} = 0.65$ mHa, yielding $M = 2417$. The error budget allocation in the qubitization approach is discussed in Appendix A 3.

For FTQC, the critical figure of merit characterizing the cost of running a quantum algorithm is the number of non-Clifford T gates. In Table III, the number of the T gates resulting from circuit synthesis and decomposition over the Clifford+ T gate set is reported for each circuit. Efficient circuit synthesis tools to compute approximations of arbitrary-angle single-qubit Z -rotations over the

Clifford+ T gate set include the well-established Solovay–Kitaev (SK) decomposition that has a T -count scaling of $\mathcal{O}(\log^c(1/\epsilon))$, with the exponent $c > 3$, and the software package gridsynth [136] based on the algorithm by Ross and Selinger [137, 138] achieving T -gate counts that are typically on the order of $4\log_2(1/\epsilon) + \mathcal{O}(\log(\log(1/\epsilon)))$, for a given allowable per-gate synthesis error ϵ . We use the latter method in our QRE studies due to its superior scaling.

As explained in Refs. [28, 50, 51], Clifford operations can be efficiently commuted to the end of the logical circuit by tracking a Clifford frame along the circuit. Some of the resulting non-Clifford gates can be merged into Clifford gates. Therefore, the process may be repeated until convergence. We call this procedure *transpilation* as explained in Appendix B 1. The outcome of transpilation is a sequence of non-Clifford gates in the form of multi-qubit $\pi/8$ Pauli rotations that must be executed using magic state injection. Therefore, the design of a fault-tolerant architecture that efficiently implements a given quantum algorithm reduces to constructing magic state factories (MSF) that can distill magic states of a target distance and fidelity at a rate on par with the fault-tolerant execution of non-Clifford gates. More details about the design of MSFs and the additional components of the layout are provided in Appendix B 2. In Table IV, we report the expected physical runtime and the number of physical qubits required for a fault-tolerant implementation of a quantum circuit when using hardware either with baseline or target parameter values, as well as for the Λ_{18} noise model (representative of a desired hardware).

To test the usefulness of parallelization and other optimization techniques, we ran smaller sample circuits through the resource estimation pipeline (see Appendix B) and estimated the resource requirements at various stages during the optimization. We found that the number of $\pi/8$ rotations before and after optimization differed only by a small factor, and that the dependency graph of the operations was nearly linear, indicating that there is no significant parallelization potential when this circuit is routed on the 2D layout. Consequently, for all circuits for which we provide resource estimates in Table III and Table IV, a single auto-correcting buffer is used in the last stage of the MSF (see Appendix B 2 for details). We note that more parallelizable circuits can be synthesized for the same quantum simulation via reorderings of the terms in the product formula. However, this can saturate the error bounds in the circuit decomposition and therefore may create nontrivial trade-offs that are interesting avenues for future research.

We plot the results of our QRE studies, including both the runtime and the number of physical qubits, in Figure 29, alongside estimates for the runtime for two classical algorithms, the variational numerically exact full configuration interaction (FCI) computation and the heuristic density matrix renormalization group (DMRG) method [139], which were calculated by extrapolating the

| Molecule Specification | | | Logical Resources | | | |
|------------------------|--------------------------------|--------------------------------------|----------------------|----------------------|----------------------|----------------------|
| | Active space | Number of orbitals, N_{orb} | $\epsilon = 1.0$ mHa | | $\epsilon = 0.1$ mHa | |
| | | | # Qubits | # T gates | # Qubits | # T gates |
| Rigor. Trotter | <i>p</i> -benzyne, HL \pm 2 | 6 | 12 | 9.5×10^9 | 12 | 4.1×10^{11} |
| | <i>p</i> -benzyne, HL \pm 6 | 14 | 28 | 9.5×10^{11} | 28 | 3.8×10^{13} |
| | <i>p</i> -benzyne, HL \pm 8 | 18 | 36 | 4.1×10^{12} | 36 | 1.6×10^{14} |
| | <i>p</i> -benzyne, HL \pm 12 | 26 | 52 | 3.6×10^{13} | 52 | 1.4×10^{15} |
| Empir. Trotter | <i>p</i> -benzyne, HL \pm 2 | 6 | 12 | 5.4×10^8 | 12 | 2.5×10^{10} |
| | <i>p</i> -benzyne, HL \pm 6 | 14 | 28 | 5.8×10^9 | 28 | 4.0×10^{11} |
| | <i>p</i> -benzyne, HL \pm 8 | 18 | 36 | 1.7×10^{10} | 36 | 8.0×10^{11} |
| | <i>p</i> -benzyne, HL \pm 12 | 26 | 52 | 4.4×10^{10} | 52 | 2.3×10^{12} |
| DF Qubitization | <i>p</i> -benzyne, HL \pm 2 | 6 | 303 | 7.8×10^6 | 341 | 9.0×10^8 |
| | <i>p</i> -benzyne, HL \pm 8 | 18 | 527 | 3.9×10^9 | 568 | 4.6×10^{10} |
| | <i>p</i> -benzyne, HL \pm 12 | 26 | 701 | 1.5×10^{10} | 748 | 1.8×10^{11} |
| | FeMoco [36] | 76 | 1792 | 1.3×10^{12} | 1972 | 1.4×10^{13} |

Table III: Logical resource estimates for electronic-structure quantum computations for two molecules, *p*-benzyne and FeMoco, and for two precisions in energy estimation: qualitatively accurate computation within a target error 1.0 mHa, and quantitatively accurate computation within a target error 0.1 mHa, respectively, using a circuit-level error budget of 0.01. We report estimates for the number of logical qubits and the number of T gates required for fault-tolerant implementations of the QPE algorithm on electronic spectra associated with various molecular active spaces for *p*-benzyne specified by HL \pm 2, 6, 8, 12 (using HL $\pm n$ to denote “HOMO $-n$ and LUMO $+n$ ”; see Appendix A 1 for an explanation of these terms) using the 6-31G basis to represent the fermionic orbitals, and the active-space model for FeMoco proposed in Ref. [36]. The sizes of the active spaces are characterized by the number of orbitals N_{orb} . Logical resources are reported for three quantum algorithmic approaches to implement QPE: a Trotterization approach based on using rigorous analytic bounds on the error resulting from the use of second-order Trotter–Suzuki approximation (Rigor. Trotter); a Trotterization approach relying on empirically obtained Trotter numbers (Empir. Trotter); and the double-factorized qubitization algorithm (DF Qubitization) of von Burg et al. [127]. The reported T -gate counts are obtained after circuit synthesis over the Clifford+ T gate set.

results of recent classical calculations [130, 140]. See Appendix D for details on this extrapolation.

This study demonstrates that ground-state energy estimation for molecules involving active spaces with orbital numbers in the range of 10 to 76 require a number of physical qubits ranging from approximately 10^6 to 10^8 and physical runtimes ranging from a few hours to several years. Both the quantum algorithm used and the hardware quality can have a significant impact on the resource requirements. On the algorithmic level, substantial space–time trade-offs can be observed. Implementations of the QPE algorithm based on Trotterization typically yield high T counts resulting in long runtimes, while the required number of qubits to run the algorithm is low, whereas implementations based on qubitization result in much lower T counts and higher qubit counts. For both algorithms, improving the hardware quality from baseline to target results in a reduced runtime and qubit count by approximately a factor of 5. Better algorithms run on better hardware can result in a reduction in runtime of up to two orders of magnitude. For example, an implementation of QPE with qubitization and target hardware results in runtime reduction by a factor of 50 compared to running QPE based on Trotterization (using empirical

bounds) on baseline hardware. We also provide results using AzureQRE in Appendix B 3.

Furthermore, we observe that, for $N_{\text{orb}} \gtrsim 25$, quantum simulations begin to outperform classical FCI computations. Linear variational post-Hartree–Fock approaches based on the FCI method are designed to provide numerically exact solutions, but their practical use is known to be limited to molecular systems with few electrons and small basis sets. Although some molecular systems are classically tractable even at scales up to 100 orbitals, in general, exact classical computations become nearly impossible beyond 25 orbitals, especially for highly correlated systems. However, powerful classical heuristic algorithms can push the quantum advantage to much greater molecular sizes. For example, the DMRG method [139] run with parallel processing on HPC units can be significantly faster than quantum simulations for molecular active spaces involving up to $N_{\text{orb}} \approx 50$ spatial orbitals, as can also be observed in Figure 29(a). Nevertheless, DMRG methods eventually become increasingly unreliable for molecular systems involving a number of spatial orbitals far beyond 50, as such systems are typically too strongly correlated, requiring calculations with very large bond dimensions, and thus intractable runtimes [35, 135].

| | | Baseline Parameter Set ($\Lambda \approx 2.34$) | | | | Target Parameter Set ($\Lambda \approx 9.3$) | | | Desired Parameter Set ($\Lambda \approx 18$) | | | |
|-------------------------|---------|---|-------------------|--------------|--------------------|--|-------------|--------------------|--|--------------|--------------------|--|
| Target error ϵ | | N_{orb} | # Phys. qubits | Phys. time | QEC code distances | # Phys. qubits | Phys. time | QEC code distances | # Phys. qubits | Phys. time | QEC code distances | |
| Rigor. Trotter | 1.0 mHa | 6 | 3.1×10^7 | 5.3 days | 15, 33, 79 87 | 1.6×10^6 | 1.3 days | 11, 29 33 | 6.6×10^5 | 23.1 hours | 23 25 | |
| | | 14 | 3.6×10^7 | 1.6 years | 17, 37, 91 99 | 2.3×10^6 | 142.2 days | 13, 33 37 | 8.6×10^5 | 111.5 days | 25 29 | |
| | | 18 | 3.8×10^7 | 7.3 years | 17, 39, 95 103 | 2.7×10^6 | 1.8 years | 13, 37 39 | 1.1×10^6 | 1.3 years | 27 29 | |
| | | 26 | 5.0×10^7 | 70.5 years | 21, 39, 107 113 | 3.5×10^6 | 15.5 years | 15, 37 39 | 1.9×10^6 | 12.3 years | 11, 29 31 | |
| | 0.1 mHa | 6 | 3.5×10^7 | 248.1 days | 17, 37, 87 95 | 2.3×10^6 | 58.2 days | 13, 33 35 | 8.4×10^5 | 44.9 days | 25 27 | |
| | | 14 | 4.7×10^7 | 76.6 years | 21, 41, 97 117 | 3.4×10^6 | 16.3 years | 15, 37 39 | 1.8×10^6 | 12.9 years | 11, 29 31 | |
| | | 18 | 4.3×10^7 | 323.2 years | 19, 41, 103 117 | 3.3×10^6 | 72.1 years | 15, 39 41 | 1.9×10^6 | 58.0 years | 11, 31 33 | |
| | | 26 | 4.7×10^7 | 2839.9 years | 19, 45, 107 119 | 3.4×10^6 | 683.4 years | 15, 41 45 | 2.6×10^6 | 501.2 years | 13, 31 33 | |
| Empir. Trotter | 1.0 mHa | 6 | 1.1×10^7 | 6.6 hours | 29, 71 81 | 1.7×10^6 | 1.5 hours | 11, 27 29 | 5.5×10^5 | 1.2 hours | 21 23 | |
| | | 14 | 3.5×10^7 | 3.1 days | 15, 33, 77 85 | 1.8×10^6 | 17.4 hours | 11, 29 31 | 7.1×10^5 | 14.1 hours | 23 25 | |
| | | 18 | 3.4×10^7 | 10.0 days | 15, 35, 79 91 | 2.1×10^6 | 2.5 days | 11, 33 35 | 7.3×10^5 | 1.8 days | 23 25 | |
| | | 26 | 3.3×10^7 | 28.4 days | 15, 35, 87 101 | 2.6×10^6 | 5.9 days | 13, 31 33 | 8.1×10^5 | 4.8 days | 23 27 | |
| | 0.1 mHa | 6 | 3.4×10^7 | 14.4 days | 15, 35, 81 89 | 2.1×10^6 | 3.4 days | 13, 29 33 | 6.6×10^5 | 2.6 days | 23 25 | |
| | | 14 | 3.5×10^7 | 239.2 days | 17, 37, 87 95 | 2.4×10^6 | 56.1 days | 13, 33 35 | 8.9×10^5 | 43.3 days | 25 27 | |
| | | 18 | 3.6×10^7 | 1.4 years | 17, 37, 89 99 | 2.3×10^6 | 120.3 days | 13, 33 37 | 8.9×10^5 | 94.3 days | 25 29 | |
| | | 26 | 3.8×10^7 | 4.2 years | 17, 39, 91 103 | 2.7×10^6 | 346.7 days | 13, 35 37 | 1.1×10^6 | 271.7 days | 27 29 | |
| DF Qubitization | 1.0 mHa | 6 | 1.5×10^7 | 5.4 minutes | 25, 61 75 | 1.6×10^6 | 1.2 minutes | 23 27 | 8.9×10^5 | 57.6 seconds | 17 21 | |
| | | 18 | 4.9×10^7 | 2.2 days | 15, 33, 75 91 | 4.0×10^6 | 12.4 hours | 11, 29 33 | 2.1×10^6 | 10.2 hours | 21 27 | |
| | | 26 | 6.0×10^7 | 10.0 days | 15, 33, 87 103 | 5.5×10^6 | 2.3 days | 11, 31 37 | 2.8×10^6 | 1.7 days | 23 27 | |
| | | 76 | 1.2×10^8 | 2.6 years | 17, 37, 93 109 | 1.5×10^7 | 234.1 days | 13, 33 43 | 8.0×10^6 | 168.8 days | 27 31 | |
| | 0.1 mHa | 6 | 2.3×10^7 | 12.6 hours | 29, 77 91 | 3.0×10^6 | 2.7 hours | 11, 27 31 | 1.4×10^6 | 2.2 hours | 21 25 | |
| | | 18 | 5.8×10^7 | 30.8 days | 15, 35, 91 105 | 5.1×10^6 | 6.5 days | 13, 31 35 | 2.6×10^6 | 5.4 days | 23 29 | |
| | | 26 | 7.9×10^7 | 132.1 days | 19, 35, 85 115 | 6.8×10^6 | 28.5 days | 13, 31 39 | 3.4×10^6 | 21.2 days | 25 29 | |
| | | 76 | 1.5×10^8 | 28.5 years | 17, 41, 99 119 | 1.8×10^7 | 6.5 years | 15, 35 43 | 9.8×10^6 | 5.0 years | 29 33 | |

Table IV: Physical resource estimates generated using the TopQAD toolkit [99] for implementing the QPE algorithm on electronic-structure quantum circuits associated with the *p*-benzynes and FeMoco molecules, for two precisions in energy estimation: qualitatively accurate computation within a target error 1.0 mHa, and quantitatively accurate computation within a target error 0.1 mHa, respectively, using a circuit-level error budget of 0.01. The corresponding logical resource requirements are reported in Table III. We report estimates for the physical wall-clock time and the number of physical qubits required for fault-tolerant implementations of the QPE algorithm for electronic spectra associated with various molecular active spaces with sizes specified by the number of orbitals N_{orb} . The data for $N_{\text{orb}} = 6, 14, 18, 26$ correspond to active space selections $\text{HL} \pm 2, 6, 8, 12$ (using $\text{HL} \pm n$ to denote “HOMO- n and LUMO+ n ”; see Appendix A 1 for an explanation of these terms) for *p*-benzynes using the 6-31G basis to represent the fermionic orbitals; the data for $N_{\text{orb}} = 76$ pertains to the active-space model for FeMoco proposed in Ref. [36]. In addition, we also report the QEC code distances that are required for running the corresponding circuits fault-tolerantly. For example, [17, 37, 93 | 109] means that the code distances $d = 17, 37$, and 93 are required for the first, second, and third magic state distillation levels, respectively, while the QEC code distance $d = 109$ is needed to encode the logical qubits of the core processor. These choices are determined by the architecture’s assembler [99] based on optimizations of the various trade-offs between the space and time costs proposed in Ref. [50]. Physical resources are reported for the same three quantum algorithmic approaches as in Table III. The associated resource requirements are reported for three hardware specifications, namely, baseline, target, and a desired hardware (i.e., the Λ_{18} model), as summarized in Table I. The results of this table are plotted in Figure 29.

While there is no such sharp transition line between what is classically tractable and intractable (which highly depends on the extent of quantum correlations in the studied systems), a quantum advantage gradually appears for orbital numbers beyond $N_{\text{orb}} \approx 50$. These insights also motivate future research, namely, developing quantum heuristic algorithms that could bring the transition to a quantum advantage down to smaller problem sizes. This naturally follows the development of classical algorithms, where the transition from the guarantees of FCI to the heuristics of DMRG greatly reduced the necessary resources.

V. TOWARD HIGH-PERFORMANCE HYBRID QUANTUM-CLASSICAL COMPUTING

Quantum computing has generated considerable interest in the high-performance computing (HPC) community as a promising extension beyond exascale systems. As accelerators within HPC infrastructures, quantum computers could perform specialized tasks rather than replacing classical computers as general-purpose systems. To reach utility-scale quantum computing, seamless integration with existing heterogeneous HPC infrastructures and the development of a full hybrid quantum-classical stack are essential.

Integrating quantum computing with high-

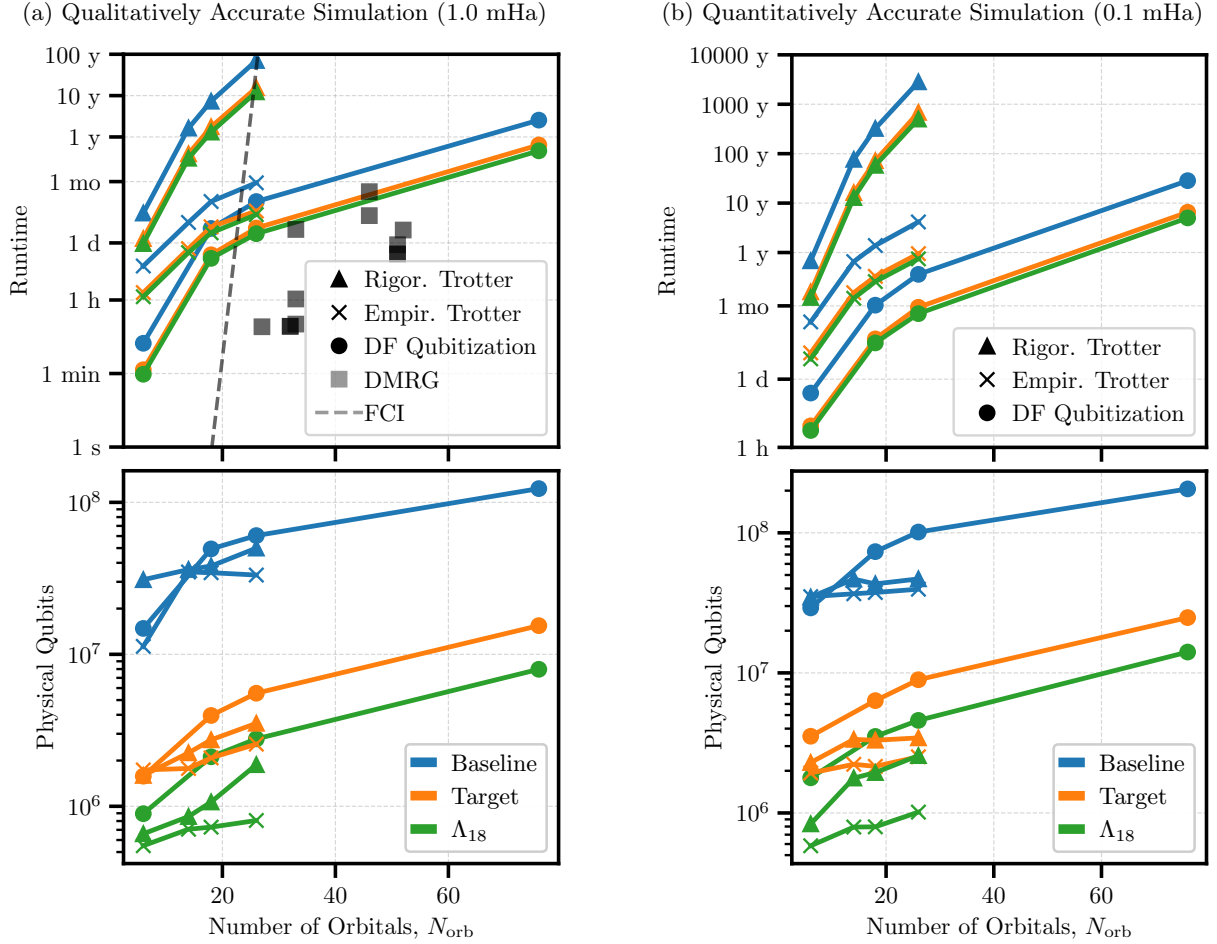


Figure 29: Physical resource estimates for electronic-structure quantum computations for two molecules, *p*-benzynes and FeMoco, and for two precisions in energy estimation: (a) qualitatively accurate simulation within a target error 1.0 mHa; and (b) quantitatively accurate simulation within a target error 0.1 mHa, respectively, using a circuit-level error budget of 0.01. For both target precisions, we report estimates for the physical wall-clock time (runtime) and the number of physical qubits required for fault-tolerant implementations of the QPE algorithm on electronic-structure quantum circuits associated with various molecular active spaces with sizes specified by the number of orbitals N_{orb} . The data for $N_{\text{orb}} = 6, 14, 18, 26$ correspond to the active-space specifications $\text{HL}\pm 2, 6, 8, 12$ (using $\text{HL}\pm n$ to denote “HOMO- n and LUMO+ n ”; see Appendix A 1 for an explanation of these terms) for *p*-benzynes using the 6-31G basis set to represent the fermionic orbitals; the data for $N_{\text{orb}} = 76$ pertains to the active-space model for FeMoco proposed in Ref. [36]. Runtime and physical qubit counts are reported for three quantum algorithms: Trotterization based on using rigorous analytic bounds on the error resulting from the use of second-order Trotter-Suzuki approximation (Rigor. Trotter), thus yielding a worst-case number of Trotter slices in approximating the Hamiltonian evolution; Trotterization relying on more-realistic, empirically obtained Trotter numbers (Empir. Trotter); and the double-factorized qubitization algorithm (DF Qubitization) of von Burg et al. [127]. Furthermore, the associated resource requirements are reported for three hardware specifications: baseline, target, and desired hardware (Λ_{18} model), as summarized in Table I. For comparison, for energy estimations within the target error 1.0 mHa, predictions of CPU times are provided for classical algorithms based on either the full configuration interaction (FCI) method or the density matrix renormalization group (DMRG) method run on a classical computer. These predictions were obtained by extrapolating the results of recent classical calculations [130, 140].

performance computing (QC-HPC) presents several challenges. On the hardware and system design fronts, key differences between quantum and classical components include physical scale, reliability, control electronics, communication bandwidth, and operational time scales. Algorithmically, the challenges involve memory access, data sharing and movement, and efficient information extraction. Some quantum algorithms lack clearly defined kernels to be off-loaded to QPUs. For certain hybrid quantum-classical algorithms, the data movement overhead associated with offloading portions to a quantum device could diminish or erase performance gains the quantum algorithm could in

theory provide. This is especially true for variational algorithms, where the quantum kernel is executed multiple times in tight interaction with a classical program. These challenges must be factored into the practical design and implementation of a hybrid quantum-HPC system. Physically co-locating classical and quantum computing resources within the same hardware node might be necessary when classical and quantum components need to exchange data with tight latency or require frequent synchronizations. To enable low-latency high-bandwidth communication, QPUs should be tightly interconnected with multiple CPUs (cores) and other accelerators such as GPUs and FPGAs, all sharing

the same system resources such as memory, cache, and high-speed interconnects.

Beyond physical integration, it is necessary to ensure the hybrid quantum-HPC system is easily programmable for the end user. Considering QPUs as accelerators and aiming for minimal changes to overall HPC program structure can mitigate the risks of complicated system development with specialized hardware. A natural solution is to integrate tools that program, compile, and execute quantum circuits into current classical HPC programming environments. Existing infrastructure for HPC (e.g. data and user management, process scheduling, control and networking) can then be leveraged for future quantum-HPC systems. For many end users, access at the HPC programming environment level will be familiar and best. More advanced users, however, may want access to the underlying physical hardware and cyber-physical control system. Different levels of abstraction in the quantum-HPC software portfolio should be harnessed for the different needs of the end users.

Figure 1 illustrates the architecture diagram for a comprehensive HPC software portfolio with extensions toward a full quantum-HPC stack. The HPE Cray Programming Environment (CPE) is a mature HPC programming system that provides software development toolchains supporting a full range of heterogeneous HPC platforms, hardware architectures, and processors. CPE provides support for multiple programming environments including HPE Cray, AMD, Intel, Nvidia, and GNU with compiler interoperability. Building on top of CPE can significantly reduce development efforts and enable rapid experimentation with different quantum SDKs (e.g., CUDA-Q[141], Qiskit[142], Cirq[143], PennyLane[144], and Classiq[145]) on available quantum and quantum-inspired accelerators as well as simulators. We can identify and target modular software capable of adapting to emerging and increasingly powerful QPU technologies, while at the same time leveraging existing NISQ QPUs as well as CPU/GPU cores for high-performance simulation. The following section describes these extensions to the HPC programming environment in further detail.

A. HPC programming environment extensions

In this section we present a software integration strategy and outline development efforts for extending quantum computing capability within the HPE Cray Programming Environment (CPE). We adopt a modular hardware/device-agnostic approach with developments for quantum programming, dispatching, and compilation within CPE. The purpose is to provide users with a unified programming environment and a full quantum-classical stack built upon existing HPC tools (compilers, libraries, parallel runtime, and process scheduling). The quantum computing capability extension includes development in three tracks:

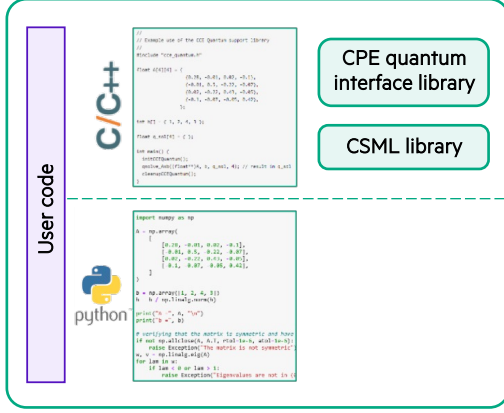
- Quantum interface library with application programming interface (API) extensions to enable seamless invocation of quantum kernels from vendor-specific quantum SDKs within HPC applications
- Quantum compiler and runtime extensions to enable performant language-level support for quantum constructs and to address the bottlenecks in compile-time with increasing circuit size
- Adaptive quantum circuit knitting hypervisor for quantum workload distribution to enable scalability with finite size (NISQ or error-corrected) quantum processors

With diverse quantum hardware including superconducting qubits, trapped ions, neutral atoms, and photonic qubits, various quantum software packages focus on different aspects of quantum computing. This may include circuit synthesis or optimizing complex design processes including qubit allocation, auxiliary qubit reuse, error mitigation, or quantum error correction. These quantum software packages are developed in different programming models and have parallelism models supporting different backends. For example, CUDA-Q supports both task-based and distributed parallel circuit execution models and provides multi-GPU, multi-node state vector and tensor network simulation backends on Nvidia GPUs[146]. PennyLane Lightning, as another example, provides state vector simulators that can be executed on both AMD and Nvidia GPUs[147].

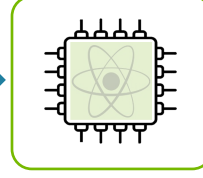
To enable seamless invocation of quantum kernels from different quantum SDKs within HPC applications developed in C/C++/Fortran, a quantum API-CPE Quantum Interface Library (CPE-QIL) is implemented in C, which can be seamlessly interfaced with Fortran applications using the `iso_c_binding` module. Application developers can use high-level, portable invocation of quantum algorithm libraries from a variety of third party SDKs within a classical HPC application in their programming language of choice. Figure 30 provides a schematic for a hybrid quantum-HPC development workflow within CPE with the quantum interface library.

Circuit synthesis and execution time in quantum computing can vary significantly based on the complexity of the algorithm and the number of qubits involved. Different SDKs offer various levels of optimization and distinct approaches to handling quantum gates, scheduling, and resource allocation, all of which impact the time and efficiency of circuit synthesis and execution. Each quantum SDK (e.g. Qiskit, CUDA-Q, etc.) has its native gate set optimized for the underlying hardware. The choice of gate set impacts synthesis complexity since some SDKs might require multiple gates to synthesize specific operations efficiently, while others may directly support them. Decomposition of high-level operations into hardware-native gate sets is usually required. For example, synthesizing a rotation or controlled operation might require

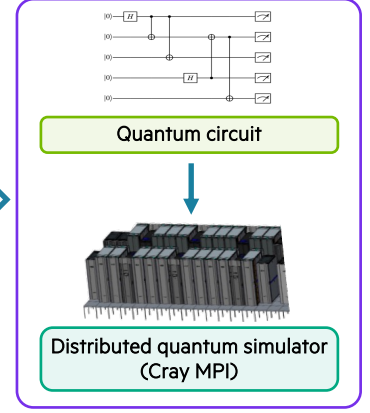
HPE Cray Programming Environment (CPE)



Circuit synthesis



Execution environment



Execution output

Figure 30: Schematic of a hybrid quantum-HPC application development workflow utilizing the CPE quantum interface library. HPC applications in Python or compiled languages (e.g. C/C++) can call different quantum SDKs (via quantum interface library) and still link with other libraries, e.g. Cray Science and Math Library (CSML). Circuit synthesis is then available from a variety of quantum SDKs—CUDA-Q, Qiskit, PennyLane, Classiq, etc. Synthesized quantum circuits can be subsequently executed on various supported quantum hardware and simulators hosted remotely on cloud or on-premise. Simulation results are sent back as return values to the HPC applications.

multiple CNOT gates and rotations, affecting both gate count and execution time. Many SDKs provide optimization levels to minimize circuit depth, gate count, or overall execution time. These optimizations directly influence both the synthesis time (by making trade-offs for synthesis overhead) and execution time on supported simulators and hardware. An example of time ranges is given in Section VIA, where circuit synthesis for Trotterization is found to take on the order of milliseconds.

Different quantum SDKs also vary in their approaches to gate scheduling, particularly for parallel gate execution across qubits (when the hardware supports it). Efficient scheduling reduces overall circuit execution time. Some SDKs optimize for parallelizable gates to reduce circuit depth, reducing execution time especially on hardware with limited coherence times. Different SDKs are tailored to different backends, with constraints on qubit connectivity, gate fidelities, and coherence times. Efficient SDKs take these constraints into account during synthesis to minimize gate count and depth based on hardware capabilities. Some SDKs are tightly coupled with specific hardware, while others support multiple hardware backends, including simulators. SDKs that are hardware-agnostic may have longer synthesis times due to added compatibility layers. See Section VIA for an example of circuit simulator execution times, which are found to be on the order of seconds.

Quantum interface library with API extensions:

The quantum interface library extension within CPE has the following advantages: 1) the ability to support and interact with a range of quantum SDKs and backends from different vendors through a standardized interface, 2) data and functionality of other software systems are available while implementation details are abstracted, facilitating efficient and secure application development, 3) support for compiled languages commonly used in mas-

sively parallel HPC applications which could allow large-scale system modeling and computation with large data sets, and 4) direct utilization of the existing HPE Cray MPI, which offers "GPU aware" MPI support and heterogeneous workload and resource managers such as Slurm and PBS (Portable Batch System) [148, 149].

The CPE quantum interface library has been utilized to deliver two hybrid quantum-HPC applications in both Python and C/C++ with quantum kernels for circuit synthesis and execution provided by Classiq's Python based quantum SDK [145]. One example consists of solving linear systems of equations with the Harrow-Hassidim-Lloyd (HHL) algorithm[150]. The solution was compared with the CPE BLAS library and showed less than 2% deviation. Another example considers the quantum approximate optimization algorithm (QAOA) for solving the MaxCut problem of partitioning a large graph into smaller sub-graphs, which can each be represented on current quantum devices. The workflow is well suited for a hybrid classical-quantum execution on supercomputers, where various sub-problems can also be solved classically if there is an advantage. Initial investigations have been conducted by HPE and Classiq and published at IPDPS24 [151]. For a simplified problem, this hybrid workload was demonstrated at ISC24 using a 20 qubit IQM quantum device accessed from the LUMI supercomputer in Finland. Understanding the latency implications in accessing a remote quantum device was one of the main goals of this investigation; the communication overhead was typically found to be on the order of seconds. This latency could be reduced for tightly integrated machines.

Figure 31 presents a detailed illustration of hybrid quantum-HPC application development and execution within CPE with the quantum interface library and HPC workload management. Within CPE, users can automatically utilize debugging and profiling tools as well as

math and communication libraries with chosen compilers. These compilers (for Fortran, C, and C++) are designed to extract maximum performance from a variety of architectures like ARM and x86-64 and devices like AMD and Nvidia GPUs. In addition, users have access to the HPE Cray Message Passing Interface (MPI), a highly scalable implementation for collective communications. Applications developed in C/C++/Fortran are compiled and linked within CPE, potentially including other libraries such as the Cray Science and Math Libraries (CSML) if the HPC application requires it. The quantum interface library is linked as a shared library during the application build process. Hybrid executables can be submitted and scheduled for parallel execution by a workload manager such as Slurm or PBS. At runtime, the interface library routes the quantum API calls from the application to the respective vendor-specific quantum SDKs with appropriate data handling (e.g. parameters and return values). When a quantum API call provided by the interface library is invoked by the application at runtime:

- Arguments to the API are converted for passing on to vendor-specific SDK;
- The interface library calls the relevant vendor-specific SDK routines to compile the quantum code into a quantum assembly language (such as OpenQASM[152], Quil[153], etc.) for a given gate-based quantum device or simulator;
- The interface library calls the vendor-specific SDK routines to execute the quantum assembly on compatible QPUs or simulators on-premise or on remote cloud-hosted resources; and
- The quantum circuit is executed (either on a quantum device or simulator) and results are converted and passed back to the application as return values of the quantum API

Quantum compiler and runtime extensions:

With increasing qubit counts, higher gate fidelity, and improved coherence times and scalability, compilation bottlenecks and latency between classical and quantum components become more apparent. When scaling to large circuit sizes with ~ 100 or more qubits, declarative frameworks struggle to handle compilation bottlenecks and latency between classical and quantum components in the program, even for NISQ systems. Leveraging classical compilation tool chains such as Clang/LLVM is crucial for developing large-scale hybrid quantum-HPC workloads, and research efforts are moving in this direction [154, 155].

To enable performant language-level support for quantum constructs and to address bottlenecks in compile-time with increasing circuit size, we have on-going efforts to build extensions on top of classical compilation tool chains. Given the diversity of pulse-level quantum instructions by different quantum hardware vendors, the

quantum assembly language OpenQASM and LLVM IR with its extension to Quantum Intermediate Representation (QIR) [156] are adopted as a middle ground for comparability with different quantum hardware and software. We leverage codegen modules to emit machine-specific native code for target architectures based on the LLVM IR/QIR produced by different quantum software front ends such as CUDA-Q. CUDA-Q provides the NVQ++ compiler for quantum kernels lowering to QIR eventually, as well as a standard library of quantum algorithmic primitives with upcoming support for quantum error correction primitives. Generated code, when linked with appropriate quantum vendor-provided runtime libraries, can be executed on target quantum devices or simulator backends in a quantum-backend-retargetable manner.

Dynamic code generation techniques are adopted to generate code at runtime based on specific characteristics of the target quantum devices. In doing so, the compiler can optimize the execution of quantum programs for different hardware generations or architectures. This low-level integration at the IR level will ensure hardware support, software compatibility, and optimal circuit compilation and execution performance for large-scale quantum-HPC workload development.

Multi-QPU workload distribution hypervisor:

The final extension to the HPC programming environment aims to tackle the problem of quantum workload distribution. For quantum computing to operate at scale, it will be necessary to efficiently parallelize over many QPUs, possibly of different hardware types, integrating them as coprocessors within an HPC framework. We enable this integration by developing a novel adaptive circuit knitting strategy serving as a hypervisor for classical and quantum compute nodes. Unlike traditional circuit knitting, this strategy uses machine learning at multiple levels to learn a quantum circuit decomposition capturing maximal quantum entanglement while enabling high-performance communication at scale. By integrating scalable message passing techniques within an adaptive adaptive circuit knitting approach, we could efficiently learn how to perform distributed quantum simulation or distributed quantum machine learning over quantum data generated by quantum processors themselves. We introduce this approach in the following section.

B. High-performance quantum workload distribution

Existing quantum processors have relatively few qubits with low gate fidelity. With the current state of technology, it is highly unlikely for NISQ devices to be scaled to tackle utility-scale problems. While proof-of-principle demonstrations of logical qubits are just starting to appear, even upcoming error-corrected quantum computers will be very small with respect to the number of logi-

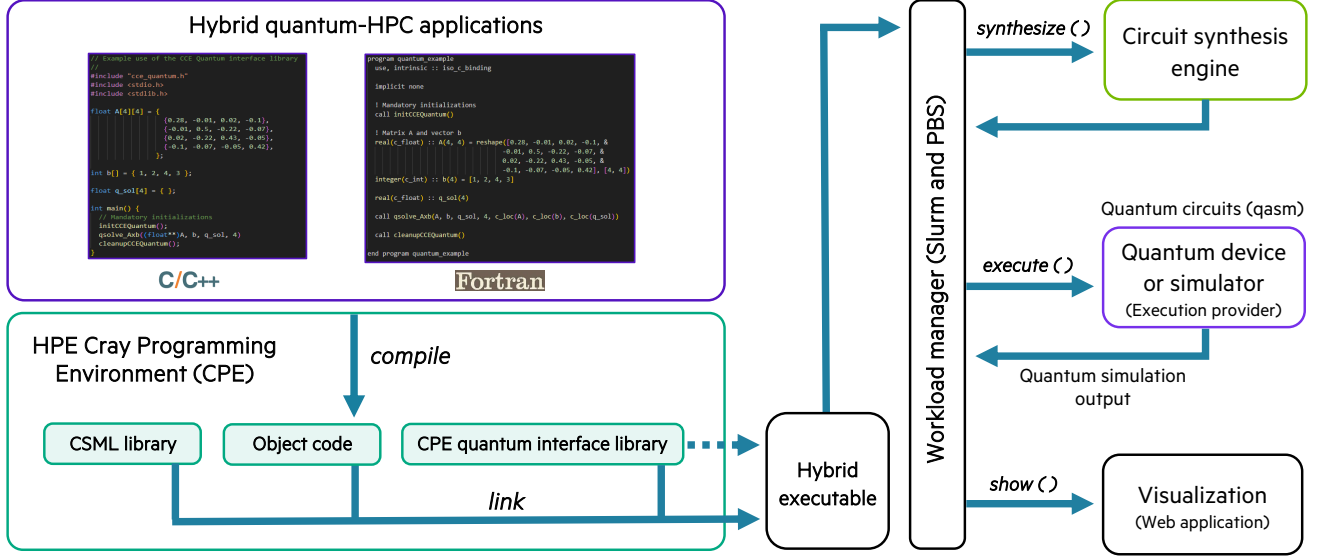


Figure 31: Development and execution of hybrid quantum-HPC applications in C/C++/Fortran through the Quantum Interface Library within the HPE Cray Programming Environment (CPE). A hybrid executable is dispatched to a workload manager (e.g. Slurm or PBS) for parallel execution. Remote visualization through web applications can be enabled with a different protocol and configurations for secure connection to clusters.

cal qubits required for utility, according to all industrial roadmaps. Recent resource estimates for FTQC indicates that the required number of physical qubits are about 0.5M-2M for quantum dynamics, 1M-6M for quantum chemistry, and 6M-30M for integer factoring [49]. At the same time, the largest quantum processors to date are on the order of hundreds of qubits, and even most optimistic roadmaps do not anticipate more than 100k qubits at a single QPU level. Consequently, to reach utility-scale quantum computing, efficiently distributing computation across multiple QPUs will be required.

Efficiently partitioning quantum systems has a rich history in quantum science [157–159]. In recent years, circuit knitting has emerged as a promising method to partition quantum circuits, the primary goal being to enable simulating large circuits on NISQ devices available today [29, 31]. In circuit knitting, a measured observable is reconstructed by sampling sub-circuits of the original circuit multiple times. This partitioning was introduced as an error mitigation mechanism by using a quasi-probability decomposition to mimic the output of a large noiseless quantum circuit by a number of smaller noisy quantum circuits [30, 160]. However, this reconstruction comes at an exponential cost in the number of samples that depends on the identity and number of gates that have been cut out [31]. While recent efforts have focused on reducing this exponential overhead [161–163], further work is necessary to demonstrate the practical advantage of circuit knitting.

To overcome the exponential classical post-processing of circuit knitting, here we introduce a family of hybrid quantum-classical algorithms for heterogeneous quantum ML/simulation. We recast this approach as Adaptive Circuit Knitting (ACK); see Figure 32. ACK can

be understood as layer-wise circuit learning [164, 165] of quantum correlations employing both feedback and feed-forward mechanisms. In this approach, we adaptively learn which essential quantum correlations to keep and which to ignore for minimizing the sampling overhead of circuit knitting. The feedback mechanism could be parallel variational quantum circuit learning, or quantum neural networks [166], over a given initial partitioning choice which can be adaptively optimized via feedforward mechanisms.

In Section VIB, we present a concrete example of ACK that consists of parallel inner-loop quantum ML and a single outer-loop classical ML enabling a distributed simulation of a quantum many-body system. To initialize, we choose a tensor network ansatz which suggests a partition of a problem onto multiple QPUs by thresholding local bond dimensions. We then variationally learn circuit parameters on all QPUs in parallel, each with circuit depth M given a local bond dimension $2M$, representing the corresponding tensor-network states but with exponentially fewer parameters. Next, we iteratively learn a new tensor network architecture via Local Operations and Classical Communications (LOCC) on each QPU and off-line classical ML. Classical communication among QPUs can be done with MPI. By using approximate tensor network representations, one can go beyond the capabilities of full state vector simulation techniques that are limited to 40-50 qubits; see Section VIA. In contrast to other approximate circuit simulation schemes such as circuit knitting proposed by IBM[31], within ACK the gate sequences and the number of qubits in each cluster could be found dynamically, i.e. on-the-fly during simulation. As an example, in Section VIB each instance of a disordered quantum many-body system is partitioned sepa-

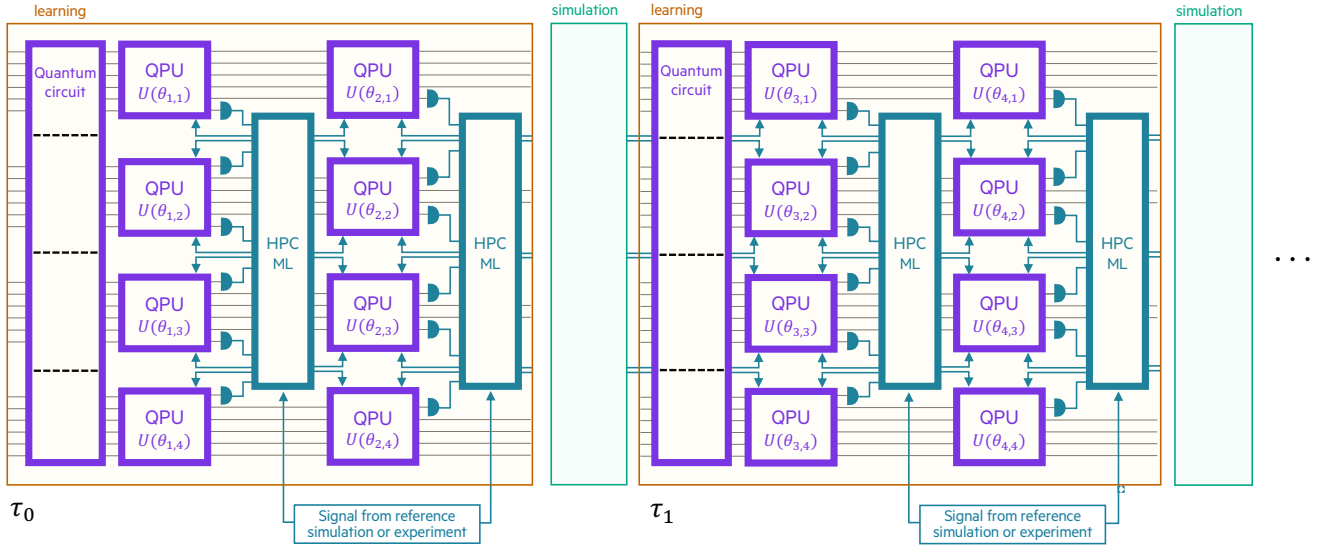


Figure 32: Adaptive circuit knitting hypervisor for distributed quantum simulation/learning: layer-wise learning of quantum correlations with feedforward mechanism could dynamically reveal an entanglement heat map. This information can guide which correlations to keep and which to ignore (circuit cuts), thus minimizing the exponential classical post-processing corresponding to circuit knitting. The overall pipeline could act as a quantum generative model which outputs quantum data that can be fed to other quantum processors.

rately. Moreover, in contrast to most alternative methods that calculate expectation values of local observables, the ACK framework can be used as a new quantum generative model. The generated quantum data can then be fed to other QPUs for further processing.

The ACK framework can act as a hypervisor that learns an efficient communication decomposition to support execution in a hybrid quantum-HPC ecosystem. As shown in Figure 1, the hypervisor can be developed within CPE while adopting a hardware-agnostic approach. HPE Cray MPI, which offers "GPU aware" MPI support, can be used to handle message passing and process management for distributed variational quantum circuit execution across multiple nodes. Integration with PennyLane and CUDA-Q can allow direct utilization of state vector simulators and tensor network simulators with multi-node/multi-GPU support as well as allowing quantum circuits to be dispatched to multiple quantum devices from different vendors.

We explore the efficiency of the ACK approach for simulating disordered quantum spin-glass system through integration with CUDA-Q within CPE, see Section VI B. Scalability and performance benchmarks can be assessed across a variety of supercomputing systems with different hardware architectures and processors. In principle, ACK could allow approximate simulations of disordered quantum many-body systems for up to thousands of qubits, but the accuracy of such approaches needs to be investigated.

C. High-performance quantum-classical workload scheduling

Efficient utilization of quantum computing resources is imperative due to their scarcity. This section explores various factors contributing to the utilization of quantum computers, especially when integrated into a multi-user environment such as an HPC cluster.

Firstly, a significant portion of quantum computer utilization is attributed to the time spent on calibration and readiness for task execution. To ensure continuous calibration, it is essential to efficiently execute the calibration graph, especially on larger scale quantum processors. In addition, it will be necessary to submit quantum benchmarking tasks to provide the quantum computer management software with information regarding its calibration status and necessary re-calibrations.

Secondly, a considerable overhead in running hybrid quantum-classical algorithms is associated with executing and loading quantum tasks, particularly when submitted by users. Quantum computing tasks exhibit significantly different timescales compared to typical HPC jobs, ranging from milliseconds to seconds. For instance, tasks involving the measurement of parameterized circuits may take only milliseconds for certain modalities like superconducting qubits, while other modalities with longer shot times may take several seconds. Note that a task duration may increase substantially when submitting a task that includes an entire iterative process. Maintaining high utilization for such tasks necessitates the implementation of an ultra-low latency interface between classical and quantum computation systems, exemplified by the DGX Quantum system. Examples of HPC jobs executing numerous short quantum tasks are

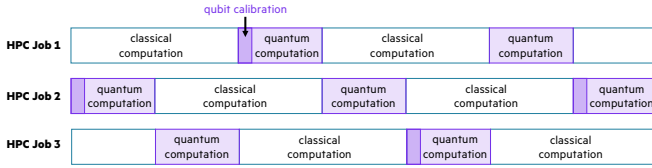


Figure 33: An example of three iterative classical-quantum algorithms concurrently executed by three different HPC jobs with an efficient scheduling of quantum computation tasks. Qubit calibration may be required during execution.

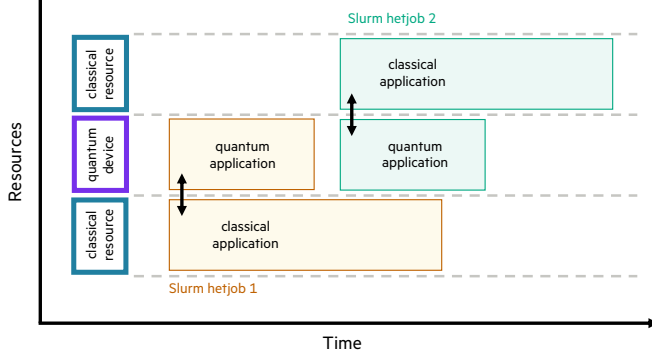


Figure 34: Schematic representation of two Slurm heterogeneous jobs requiring a quantum device that is exposed as a compute node. As soon as the quantum device is no longer needed by the first heterogeneous job it can be released while the classical part continues to run. The second heterogeneous job can then start using the quantum device.

hybrid algorithms with iterative quantum and classical coprocessing. Such hybrid algorithms perform an iterative process of measuring parametric circuits and subsequently calculating the next set of parameters based on the obtained measured results. During the execution of a single algorithm, the quantum computer often remains idle while HPC nodes retrieve measurement results, perform calculations, and submit new quantum tasks. This idle time between quantum tasks within the same HPC job can be particularly prolonged in an interactive workflow, where user actions trigger the submission of subsequent quantum tasks.

To optimize quantum computer utilization, it is essential to schedule tasks from different algorithms and different HPC jobs concurrently, thereby minimizing idle periods while the algorithm performs the classical computations, as shown in Figure 33.

In the case where an optimal scheduling of classical and quantum resources is not possible or not necessary, block allocation of a quantum device by a single user is legitimate, especially in the presence of several quantum resources. This can be achieved by workload managers (WLM/Scheduler) such as Slurm that are already present in HPC environments (see Section V A). With a workload manager, quantum resources could be exposed as regular nodes encapsulated in a partition.

Figure 33 represents an example scenario in which one or more classical applications running on multiple nodes share the qubits in all/none fashion. This exam-

ple originates from a demonstration at ISC23 where the scheduling and inter-process communication was accomplished via the multiple data multiple program (MPMD) paradigm in Slurm. This MPMD model is simple, but has the drawback of blocking a quantum device for the duration of the classical application, potentially wasting resources. A reduction of this idle time can be achieved through the Slurm support for heterogeneous jobs (`hetjobs`) to split a job across differing hardware. A simple scenario consists of two heterogeneous jobs, each requiring classical and quantum computing resources. As is typical in HPC, the two jobs are submitted to a queue. Once resources are available, both the classical and quantum parts of the job begin. At a crucial point in the execution, a synchronization requires the classical part to wait on results from its quantum counterpart. Once the quantum computation is finished, the resource is freed and then immediately consumed by the quantum part of the next hybrid job, which has been waiting in the queue for the resource to become available. The contemporary start of the quantum and classical computations is not always guaranteed.

In order to enable the efficient fine-grained scheduling mentioned above, more intelligent adaptive and heterogeneous task scheduling algorithms need to be considered. These must account for quantum resource availability and dynamically adjust task assignments based on runtime feedback. A customized Slurm plugin to discretize quantum workloads into pulse-level tasks needs to be developed. It will also be necessary to adopt a partition-based approach which allows the application to split available qubits based on the underlying quantum resources. A WLM for hybrid systems should enable the allocation of available qubits among users across different nodes containing QPUs. In the following section (Section VI) we provide a few examples of high-performance distributed quantum simulations for studying dynamics of quantum spin-glasses near quantum phase transitions. These examples could provide useful testbeds for developing high-performance hybrid workload scheduling.

VI. A NEAR-TERM APPLICATION: DISTRIBUTED QUANTUM SIMULATION

In the near term, quantum computers will continue to have relatively few qubits with low gate fidelity. During this time, classical simulators of quantum computers, especially those optimized for performance on HPC systems, are important for prototyping, benchmarking, and quantum algorithm development. In this section we present two examples that highlight the importance of HPC systems in this development process, both targeting near-term applications in condensed matter physics: dynamical quantum phase transitions in 2D transverse-field (quantum) Ising models, and strongly-ordered quantum spin glasses. In both cases, we discuss the importance of distributed quantum simulation, either through classical

HPC or algorithms for quantum workload distribution such as adaptive circuit knitting, and outline possible directions for new research in this important area.

A. Multi-GPU: Dynamical quantum phase transitions of 2D transverse-field Ising models

In this section we demonstrate the use of the multi-node, GPU-accelerated CUDA-Q[146] state vector simulator to study exotic phenomena in quantum materials. Studying such phenomena can help us better understand materials properties or even control physical/chemical systems in their condensed phase, aiding in the design of new materials, e.g. for quantum sensors. This work, carried out in collaboration with Nvidia, shows how CUDA-Q can compile and execute distributed quantum circuit simulations on HPC systems.

The transverse-field Ising model (TFIM), the quantum analog of the classical Ising model, is a well-studied system in the condensed-matter physics community. It describes a lattice of N spins with nearest-neighbor interactions in the presence of an external magnetic field, with a Hamiltonian given by

$$H = -J \sum_{\langle i,j \rangle} \sigma_i^z \sigma_j^z - g \sum_{i=1}^N \sigma_i^x \quad (9)$$

where $\langle i,j \rangle$ denotes all nearest-neighbor pairs in the lattice, σ^z and σ^x are the Pauli Z and X matrices, respectively, and J and g are parameters that control the nearest-neighbor coupling and transverse-field strengths, respectively. Despite its simplicity, the TFIM can exhibit complex quantum phenomena that could be difficult to simulate classically. This is especially true for 2D spin lattices, which are thought to be beyond the capabilities of approximate simulation methods like matrix product states (MPS) and quantum Monte Carlo (QMC). Simulating many-body quantum systems (like the TFIM) beyond 1D is an open challenge, especially for non-equilibrium or excited-state properties. One such property is dynamical quantum phase transitions (DQPT), which are non-equilibrium phase transitions of quantum systems in time [167].

Studying these systems is a promising use case for circuit-based quantum computers because their continuous time evolution, given by $|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle$, can be simulated with a discrete-time digital circuit via the Trotter procedure [168, 169]. In principle, this discretization enables scaling up such simulations on fault-tolerant quantum computers (see Section IV for a more thorough discussion on this topic). For Hamiltonians like the TFIM that can be written as the sum of local terms, the time-evolved state can be approximated by

$$|\psi(t)\rangle \approx \left(\prod_j e^{-ia_j H_j t/r} \right)^r |\psi(0)\rangle \quad (10)$$

where t/r is the timestep in the evolution. For large enough r , the product of matrix exponentials is a reasonable approximation for the sum of matrix exponentials. We find that a first-order Trotterization, e.g. $e^{-i(A+B)t} \approx (e^{-iAt/r} e^{-iBt/r})^r$ is sufficient for accurate simulations of DQPTs in the TFIM.

A dynamical quantum phase transition occurs as a result of "quenching" a quantum system. The initial quantum state $|\psi(0)\rangle$ represents the ground state of Hamiltonian H_0 . (For example, a state with all spins up like the one shown in Figure 35 is the ground state of the Hamiltonian with $J_0 = 1.0$, $g_0 = 0.0$.) The time-evolution is then carried out under a different Hamiltonian H . This forces the system to undergo a rapid phase transition in time. The quantity of interest when studying DQPTs is the Loschmidt amplitude $\mathcal{G}(t)$, which is the overlap of the time-evolved quantum state with an initial state: $\mathcal{G}(t) = \langle \psi(0) | \psi(t) \rangle = \langle \psi(0) | e^{-iHt} | \psi(0) \rangle$. The Loschmidt echo $\mathcal{L}(t)$ is the probability associated with the amplitude: $\mathcal{L}(t) = |\mathcal{G}(t)|^2$. We can identify DQPTs by tracking the rate function, given by

$$\lambda(t) = - \lim_{N \rightarrow \infty} \frac{1}{N} \log \mathcal{L}(t) \quad (11)$$

where N is the number of qubits. A DQPT occurs at critical time t_c where there is a non-analytical peak in $\lambda(t)$.

While there have been recent demonstrations simulating DQPTs on both quantum devices (a subset of qubits in a 22-qubit superconducting chip [170] and 53 qubits in a trapped ion experiment [171]) as well as using numerical classical simulators [167], these studies have been limited to 1D. Understanding phase transitions in 2D is likely key for designing real devices and materials. CUDA-Q's `cuStateVec` backend, which represents the entire 2^N state vector and can capture maximum entanglement, enables accurately simulating 2D systems and computing the rate function $\lambda(t)$ throughout the time-evolution.

We performed several simulations of 2D spin lattices on various compute configurations ranging from a single CPU to 512 GPUs across 128 nodes. Figure 35 shows a DQPT discovered during the simulation of the largest system studied, an 8x5 spin lattice (40 qubits). During many of phase transitions simulated, the entanglement entropy in the quantum system grows to near its maximum value, making them difficult to simulate classically with tensor network techniques.

Circuit synthesis time for one Trotter time step ranges from 0.2-0.4ms for 5-10 qubits to 2-5ms for 25-33 qubits on a single A100 GPU. Saving the previous state in GPU memory instead of re-simulating all previous operations at given time step will drastically reduce the circuit synthesis time, keeping it flat with increasing circuit size as time step increases.

Figure 36 shows the performance comparisons for multi-threaded CPU, single A100 GPU, and multiple A100 nodes. Circuit execution time for one Trotter time step ranges from 0.04s-4s for 20-25 qubits on a single

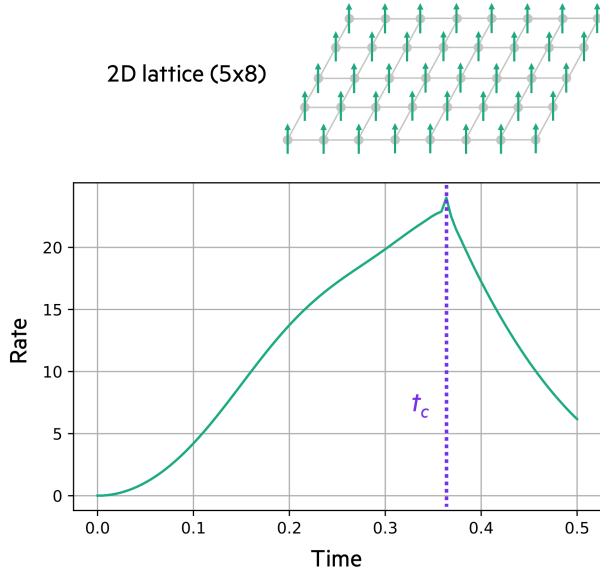


Figure 35: Dynamical quantum phase transition observed at t_c during time-evolution simulation of a 40-qubit 2D Ising model with $J = 1.0$, quenching from $g_0 = 0.0$ to $g = 5.0$. Simulation comprised 100 timesteps executed on 512 A100 GPUs across 128 nodes on Perlmutter.

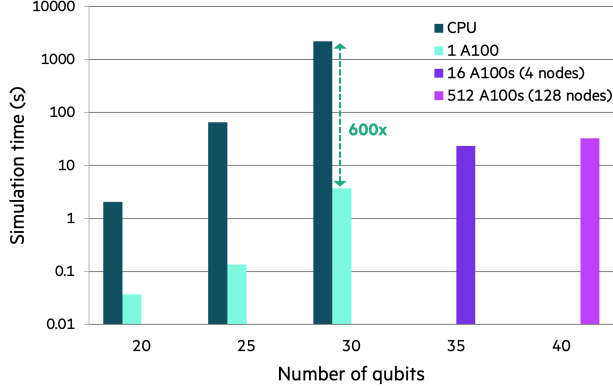


Figure 36: Performance of CUDA-Q simulation using multi-threaded CPU, single GPU, and multi-GPU backends. Systems simulated are 2D lattices: 5x4, 5x5, 5x6, 5x7 and 5x8 qubits. Simulation time reported is for one timestep in the time-evolution circuit (100 total timesteps).

A100 GPU to 23s-33s for 35 qubits (distributed across 64 A100s) and 40 qubits (distributed across 512 A100 GPUs). For 20 to 30-qubit simulations, one A100 provided a 600x speedup over a multi-threaded CPU. Beyond 30 qubits, the exponential scaling of quantum simulation quickly outstrips the capabilities of a single processor, but scaling to 40 qubits was possible by distributing the simulation across 128 nodes (512 A100 GPUs) on the Perlmutter supercomputer. The 40-qubit simulation took one hour, nearly two orders of magnitude faster than a CPU simulation on 30 qubits. The performance results on these larger qubit systems highlight the multi-node parallel efficiency of both the software and hardware.

Simulations of many-body quantum systems like the

ones shown here are important for the near-term benchmarking of quantum computers—a high-performance state vector simulator can enable the accurate study of DQPTs in 2D spin lattices up to 40 qubits, a task currently beyond the capabilities approximate methods. This framework could be used to study other quantum systems as well as study the effects of noise, a crucial element for the development of near-term quantum devices. However, state vector simulations much beyond 40 qubits are out of reach even for the most powerful supercomputers. Approximate methods such as tensor network techniques become crucial. In the following section (Section VIB), we provide an example of applying tensor network techniques to an important problem for scaling quantum computing: distributing quantum workloads.

B. Multi-QPU: Strongly disordered quantum spin glasses

Section VB introduced adaptive circuit knitting (ACK) approaches for quantum workload distribution that aim to overcome the exponential overhead of circuit knitting. In this section, we present a concrete example of ACK which decreases sampling overhead of circuit knitting by cutting in locations that minimize entanglement between partitions. We describe the method and demonstrate its application simulating the dynamics of quantum spin chains.

This particular ACK method draws from tensor network (TN) approaches developed in the quantum physics and quantum chemistry communities [157, 158]. Tensor networks represent quantum states in a compressed form, and can provide structure to characterize entanglement patterns. In the context of quantum circuits, a TN can be efficiently expressed as circuit of linear depth as was shown by Lin et. al [172] for matrix product states (MPS), a type of TN widely used to study 1D quantum systems. Combining the structure of TNs with linear depth quantum circuits is the basis for this ACK method.

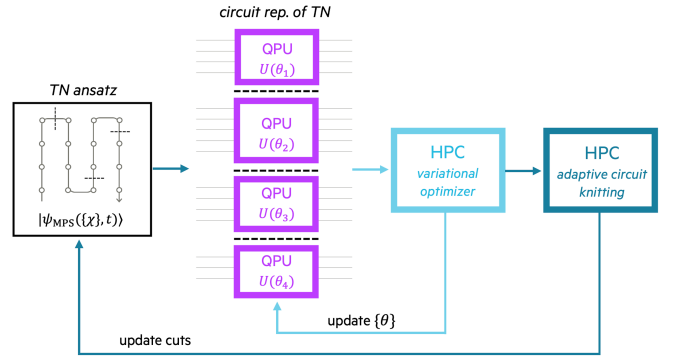


Figure 37: Schematic of an adaptive circuit knitting method using tensor networks. In the inner loop, a variational optimizer finds circuit parameters for partitions of a quantum system (based on a tensor network) in parallel. In the outer loop, an adaptive procedure finds cuts which minimize entanglement between partitions. After the best cuts are found, observables are reconstructed via circuit knitting.

Figure 37 provides a schematic for this adaptive circuit knitting method. It begins with a TN representation of a quantum state (in the figure, an MPS for a 2D spin lattice). The TN is partitioned into N sub-networks, drawn here with $N = 4$. In an inner loop, for each sub-network the variational procedure outlined by Lin et al. [172] is followed and gates $U(\theta_N)$ are optimized for an efficient circuit representation of each sub-network. Following this optimization, in the outer loop entanglement measures (e.g., von Neumann entropy) are computed and an entropy heatmap among the qubits is constructed. Although this heatmap is partial (we do not have entropy measures at the cuts between partitions), the information available is used to update cuts to locations with low entanglement. With new cuts on the following iterations, the blind spots are revealed. The adaptive outer loop exits when entanglement between partitions is minimized. The inner loop can be parallelized since each sub-network is independent, and both the inner and outer loop can be executed using classical HPC. Once optimal partitions have been found, a measured observable can be reconstructed from the sub-circuits via circuit knitting [31]. As we show below, cutting gates at locations of minimal entanglement can substantially lower the classical overhead of circuit knitting.

Simulating quantum systems for materials science or quantum chemistry is one of the most promising applications for quantum computers. Spin-lattice systems are well-studied in materials science, and despite their simplicity can exhibit complex quantum phenomena that are difficult to simulate classically, e.g. the dynamical quantum phase transitions discussed in Sec VIA. As a prototype system, we apply this ACK method to simulating the non-equilibrium dynamics of a strongly-disordered spin chain evolving under an Ising model with transverse and longitudinal fields given by the Hamiltonian

$$H = - \sum_{i=1}^{N-1} J_{i,i+1} \sigma_i^z \sigma_{i+1}^z - \sum_{i=1}^N g_i \sigma_i^x - \sum_{i=1}^N h_i \sigma_i^z \quad (12)$$

where σ^z and σ^x are the Pauli Z and X matrices, respectively, i indexes the lattice site, and J s, g s, and h s are real-valued parameters. We study strongly disordered systems (where parameters are varied at each lattice site) because they are important for understanding exotic states of matter, they can be difficult to study, and because they lead to many-body localization effects which we suspect could be exploited for more-efficient simulation.

Figure 38 provides a summary of the results for an ensemble of 32-qubit spin chains, each time-evolving under a Hamiltonian with different parameters chosen randomly from a uniform distribution on $[-1, 1]$ (excluding 0). For each spin chain, circuit optimization was carried out at eight timesteps throughout the dynamics. We partition each system into two sub-circuits and compare the overhead cost of circuit knitting for a naïve cut in the middle of the chain (a “load-balanced” choice) versus a cut recommended by the entropy heatmap from

the adaptive algorithm. Figure 38(a) gives a schematic for a single instance on a smaller 20-qubit system. Figure 38(b) shows the distribution of overheads for reconstructing a magnetization observable via circuit knitting for the adaptive and load-balanced cuts. Both cuts achieve similar accuracy in the observable, but in most cases the adaptive cut results in a much lower overhead—the green distribution is distinctly shifted to the left. On a case-by-case basis, the median reduction in cost was $15\times$, while the 75th and 95th percentiles were $59\times$ and $450\times$, respectively. While not shown in this figure, the gap between adaptive and load-balanced widens during the later timesteps, indicating that for longer simulations the benefits of adaptive circuit knitting will increase. As in Section VIA, the `cuStateVec` simulator was employed to enable performant execution of the 32-qubit simulations.

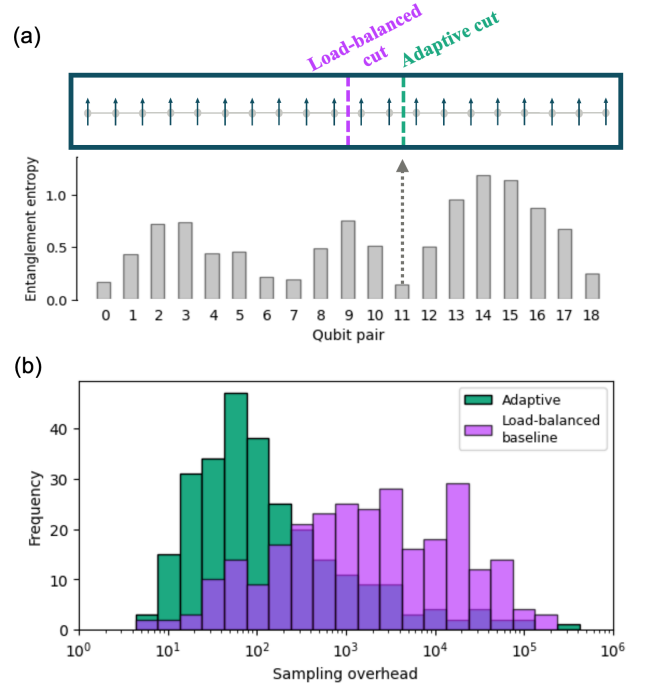


Figure 38: (a) Example of a 20-qubit spin chain where the adaptive cut is chosen at the minimum entanglement entropy. (b) Histogram of sampling overheads resulting from adaptive and load-balanced baseline cuts for an ensemble of 32-qubit strongly-disordered spin chains.

Such initial results showing improvements in overhead of one to two orders of magnitude suggest that this ACK method could be used for efficiently partitioning quantum circuits for near-term applications simulating condensed-matter systems. However, more study is necessary to demonstrate the practical utility of such a method. Future work will include investigating fast entanglement measures, exploring more sophisticated convergence criteria, and studying 2D systems with higher order TN techniques, as 2D systems are where many classical methods underperform and quantum computers could likely provide the largest advantage (See Section VB for a more general discussion of ACK techniques.) A high-

performance implementation in CUDA-Q could enable fast prototyping of these methods for circuit sizes large enough to capture interesting physics.

VII. TOWARD HETEROGENEOUS QUANTUM AND PROBABILISTIC COMPUTING

Parallel to the development of NISQ devices and FTQCs architectures, an emerging trend in computing has been to build quantum-inspired accelerators for combinatorial optimization and sampling problems. A notable example is the notion of a probabilistic computer (p-computer) with probabilistic bits (p-bit) [173–175]. It has been shown that networks of *hardware* p-bits natively represent a wide class of probabilistic algorithms typically implemented in *software*, with significant energy and performance benefits [175–179]. From an HPC perspective, domain-specific probabilistic computers can accelerate hard optimization and sampling tasks by orders of magnitude. These accelerators can be integrated in a distributed fashion within heterogeneous hybrid quantum–classical hardware architectures, which we discuss further in Section VIII.

A. Probabilistic computing with intrinsic higher-order interactions

p-computers implement a Markov chain Monte Carlo algorithm called Gibbs sampling, achieved by a stochastic activation and a local field calculation, given by:

$$m_i = \text{sgn}(\tanh(\beta I_i) - r_U), \quad (13)$$

$$I_i = \sum_j J_{ij} m_j + h_i, \quad (14)$$

where m_i represents the bipolar p-bit state (± 1), r_U is a uniform random number between $(-1, +1)$ and $[J], \{h\}$ are the weights and biases for a given problem and β is the inverse temperature.

There are two immediate generalizations possible: (a) p-bits can be extended to have multiple states. These Potts spins have also been implemented in hardware [180] and shown to have better embedding than simple p-bits for certain optimization problems such as graph coloring. (b) The graph connectivity defined by J_{ij} can be generalized to *hypergraphs* where the local field equation becomes (e.g., for $k=4$ -local interactions)

$$I_i = \sum_j J_{ij} m_j + \sum_{j < k} J_{ijk} m_j m_k + \sum_{j < k < l} J_{ijkl} m_j m_k m_l, \quad (15)$$

where J_{ijk} and J_{ijkl} denote the interaction coefficients for three and four-local interactions respectively. These can be generally extended to any k -local interactions (Ref. [179] demonstrates an implementation with $k=3$

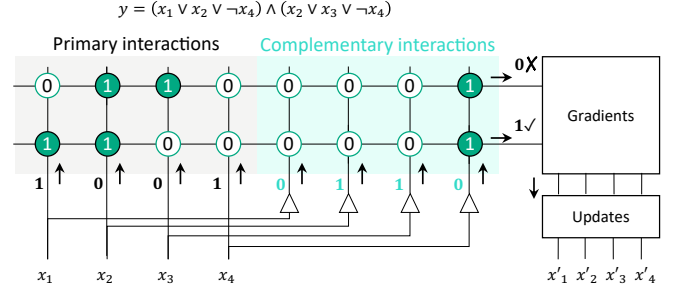


Figure 39: Accelerating k -local interactions with in-memory computing. The interactions in a SAT formula can be evaluated by computing the Hamming distance between the input x and each one of the clauses, mapped such as a Hamming distance of 0 corresponds to a clause violation. Two coupling arrays are used to represent primary and complementary interactions. A positive literal in a clause, such as x_1 in the first clause, is mapped as 1|0 in the primary coupling array and complementary coupling array, respectively; negative literal, such as $\neg x_4$ in the first clause is mapped as 0|1; and a non-member literal, such as x_3 in the first clause, as a 0|0. After the interactions are computed, the Hamming distance output can be used directly to compute high order gradients.

to solve the XORSAT problem). The binary nature of p-bits significantly eases the implementation of k -local interactions. Such k -local interactions greatly reduce model embedding and complexity. We are not aware of any programmable multi-qubits entanglement for $k > 2$ in quantum computers but for probabilistic computation such hypergraphs can be constructed rather easily.

Recently, higher order k -local interactions architecture without limits or scaling dependence on the order k based on in-memory computing have been presented [181, 182]. In the proposed approach [181, 182], the interaction between variables is computed before computing the gradients by encoding the clause member variables in an interaction matrix. In the case of k -SAT problem with N variables and M clauses, a $2 \times N \times M$ matrix is used for embedding interaction, where the factor 2 accounts for a primary interaction matrix and its complementary as shown in Figure 39. Member variables in clauses are encoded with $[1|0]$ and $[0|1]$ for x and $\neg x$, respectively. A non-member variable is encoded as $[0|0]$. Considering the first clause of the 3-SAT formula y in Figure 39 with $N=4$, $y_i = (x_1 \vee x_2 \vee \neg x_4)$, its encoding in the interaction matrix I is $I_i = [1, 1, 0, 0|0, 0, 0, 1]$. Given for example an input $x = [1, 0, 0, 1]$, if the Hamming distance between $x' = [x|\neg x]$ and I_j , $\delta(x', I_j) > 0$ the clause is satisfied. In the example $\delta(x', I_j) = 1$, meaning the clause is satisfied but only one literal (x_1) is positive, thus by flipping it the clause becomes unsatisfied.

Note that compared to traditional Quadratic Unconstrained Binary Optimization (QUBO) mapping, such as the one used in conventional Ising machines, the higher-order interactions allow for a native embedding of the problem, without the need for auxiliary variables. Native mapping leads to improved convergence, no introduction of artificial local minima and settle points due to auxiliary variables [183], and reduced hardware resource utilization by $O(k^2)$.

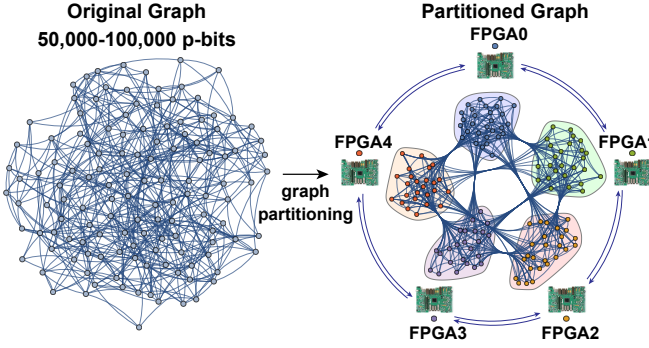


Figure 40: Distributed p-computers: a single large graph is partitioned into multiple smaller subgraphs to distribute it across multiple processing elements (PE) in the form of FPGAs/GPUs/TPUs. A graph partitioning tool is used to ensure minimum cut across the subgraphs to minimize communication overheads. Preliminary results (in preparation) show over 1000 probabilistic flips per *nanosecond* can be taken in these distributed systems, with about 2 orders of magnitude improvement over single GPU/TPU implementations.

B. Hardware implementation of p-computers

Physical implementation of p-computers includes a wide range of choices from noisy materials to analog and digital CMOS. State-of-the-art p-computers demonstrate nanodevice (magnetic tunnel junction, MTJ) based prototypes [178, 184], where the natural noise of the stochastic MTJ provides computational resources to solve a small-scale problem. This prototype has shown the promise of magnetic RAM technology, as a scalable pathway to build energy-efficient p-computers. Magnetic memory industry has achieved Gigabit densities of magnetic tunnel junctions embedded with CMOS transistors in monolithic integrated circuits [185]. Repurposing these MRAM chips so that their *stable* MTJs become *unstable* (low-barrier) could lead to dedicated probabilistic computers with tens of millions of integrated p-bits. Before an integrated p-computer using millions of stochastic magnetic MTJs, however, digital emulators of p-bits using powerful CMOS-based Field Programmable Gate Arrays (FPGA) have been used to investigate the architectural and algorithmic performance of p-computers at large-scale [175–179, 186]. Even single FPGA-based implementations of p-computers have shown competitive performance against the state-of-the art [179].

C. Scaling up p-computers: a distributed approach

Very often the sizes of single processing elements (PE), in the form of FPGAs/GPUs/TPUs, are not large enough to encode practical problem sizes containing thousands to millions of variables. To get around this problem, one solution is to design *distributed* architectures where a large problem is partitioned into smaller subgraphs housed in distinct PEs (Figure 40). Preliminary results show that as long as the communication links are faster than individual p-bit clocks, distributed probabilistic computers

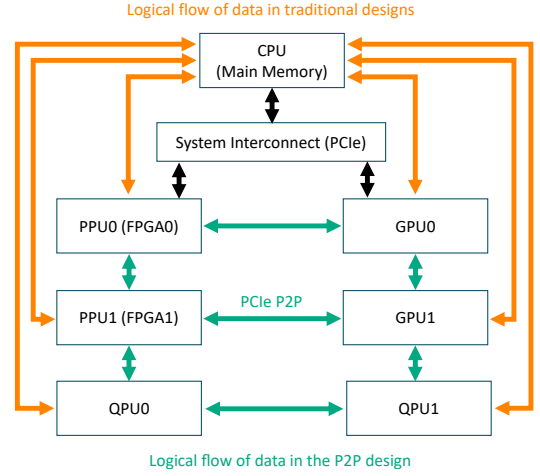


Figure 41: Heterogeneous system architecture block diagram with conventional approach (orange) and the decentralized peer-to-peer (p2p) communications. (green). Removing the required communication to CPU and main memory significantly improves the performance allowing direct communication between multiple accelerators. Heterogenous systems including conventional digital accelerators (GPU), quantum processing units (QPU) and probabilistic processing units (PPU) can be built to potentially enable novel classes of heterogeneous classical/probabilistic/quantum algorithms.

can create the “illusion” [187] of a single PE that can house a much larger graph. Sampling rates over 1000 flips per *nanosecond* are feasible (in preparation), boding well for hard combinatorial optimization and sampling problems.

More generally, many workloads in probabilistic AI models, such as modern Energy-based Models (EBM) [177, 188], involve several linear algebra operations, mainly as matrix multiplies. Thus, to scale up probabilistic architectures a heterogeneous approach involving traditional digital accelerators (e.g., GPU) and p-computers is desirable. The GPU can be used for a large part of the forward operation, such as computing embeddings, the gradients, and loss optimization while the p-computers (implemented for example on an FPGA) can essentially implement the stochastic neuron operation. Figure 41 shows the example of a heterogenous system architecture with multiple GPUs and FPGAs that can be used for scaling up to a very large number of p-bits (e.g. >1B) making it able to train and infer next-generation EBMs.

Two natural concerns arise. First, given the EBM consists of a lot of matrix operation, careful system-level profiling should be performed to assess if the time for sampling is the one to optimize by Amdahl’s law. Note that in the generalized case of a fully connected model, by scaling linearly the number of neurons the number of synapses scales quadratically so at each step of an exponentially long sampling process, linear operations (i.e. activations) are performed on neurons and quadratic operations (i.e. matrix multiplies) on synapses.

Second, their PCIe communication might bottleneck the heterogeneous approach with several, e.g., GPU2GPU, FPGA2FPGA, and GPU2FPGA calls. In traditional architecture, every time a GPU communi-

cates with the FPGA it should access the main memory through PCIe, as shown in the orange flow of Figure 41 leading to significant overhead due to back-and-forth communications. Architectures with Peer2Peer (P2P) communication [189] and disaggregated memory [190, 191] could potentially limit such bottleneck as shown in the green logical flow of Figure 41, minimizing this back-and-forth access through CPU where FPGA-GPU can directly talk to each other through PCIe avoiding main memory access on CPU [192].

D. Quantum-assisted probabilistic computing with custom-design accelerators

There are three complementary perspectives that we can envision for the interplay of quantum fluctuations and thermal fluctuations in a probabilistic computing framework: within the problem space, the algorithm space, and the solution space. Within the problem space, as we described in Section VII C (see Figure 40), we can partition a general dense graph with higher order interactions by sparsification techniques. During the procedure we are iteratively creating conditional probability distributions; i.e., by freezing or clamping a subset of variables, and sampling over the rest of variables residing on a smaller and lower dimensional subgraph. This technique can be used to reduce both the size and complexity of the problem such that it can be easily embedded on finite-size low-dimensional quantum accelerators/solvers. These quantum solvers could be either analog, based on quantum annealing [193, 194], or digital, based on Quantum Approximation Optimization Algorithm (QAOA) [69, 70]. Larger, denser, and/or highly structured subgraphs can be sampled via p-computers and the outputs can be used as boundary conditions (e.g., local fields for Ising machines) for potentially quantum-prone subgraphs iteratively. Within the algorithm space, we can understand the role of quantum solvers is to provide hot starts/seeds for classical probabilistic framework and vice versa. This was originally introduced in the context of creating non-trivial initial seeds for reverse quantum annealing or MCMC sampling enabling quantum-assisted parallel tempering [195] or quantum-assisted genetic algorithms [196].

Within the configuration space, we can see the role of quantum fluctuations as a new mechanism to navigate in the saddle regions or regions with shattered configurations space with many unstructured shallow barriers. Such effective quantum walks could in principle lead to a quadratic speedup for diffusing in the configuration space over a classical walk [67]. It should be noted that these three perspectives are not mutually exclusive, for example in the context of non-equilibrium non-local Monte Carlo algorithm developed [197], one can discover backbone or cores of frozen or rigid variables in a configuration space near a phase transition (e.g., for k -SAT problems near a computational SAT/UNSAT phase tran-

sition). The backbones with a higher degree of connectivity could be sampled with classical probabilistic accelerators to induce large Hamming distance exploration ($O(N)$) at scale. Then the new coordinates can be passed to quantum accelerators to create smaller-scale (in Hamming distance) nonlocal explorations over regions with significant entropic barriers or shallower energy barriers that would be prone to quantum tunneling on finite-size quantum processors. On the other hand, the new local minima found by a quantum processor can be improved by classical fluctuations, especially those induced by quantum many-body localization effects [198].

This hybrid algorithm can be incorporated within the heterogeneous HPC computing platforms, with p2p communications among various nodes including CPUs, GPUs, FPGAs, QPUs, or other custom design accelerators, see Figure 41. In some implementations, both quantum and classical processors can be placed on the same chip to get additional performance benefits [199] (e.g., in hybrid quantum and classical superconducting processors). This quantum-probabilistic framework can enhance the diffusion in configuration and improve the quality and diversity of solutions [200, 201] given a time or energy budget, as new basins of attractions could be found orders of magnitude faster and more energy efficiently than using either probabilistic or quantum accelerators alone.

VIII. DISCUSSIONS

A. Supply chain management toward utility-scale

A utility-scale quantum computer should be characterized as a machine whose computational value surpasses its total cost [6]. This cost encompasses not only manufacturing and operational expenses but also the amortized research and development investments. Achieving the utility scale is a crucial milestone in quantum computing, signifying the point at which these advanced systems become economically viable for practical applications. A key premise of our position is that leveraging the existing semiconductor supply chain can help amortize the cost of research and manufacturing. To further this amortization, it is also important to ensure reusing quantum modules and manufacturing technologies for as many quantum applications as possible, with significant computational demands.

Analysis of the semiconductor supply chain raises two issues: 1) Is the supply chain competitive enough to reduce costs while avoiding dependence on a single source or foreign-controlled manufacturing? 2) What existing semiconductor research programs can be leveraged to reduce the research cost of a utility-scale quantum computer within next 10 years? From a manufacturing cost perspective, moving to semiconductor compatible fabrication processes is essential to drive down costs. A

number of 300-mm advanced fabrication facilities are being built in the U.S. through public-private partnerships (e.g., the US CHIPS Act) which will reduce reliance on single-source/foreign manufacturers. Although this approach would require upgrading the tooling of the semiconductor foundries, the upgrades are a fraction of the cost of building a state-of-the-art 300-mm foundry. This in turn will drive down the cost of the various components needed to build a quantum supercomputer. Furthermore, existing research fabs at locations such as Applied Materials and NY CREATES could act as a stopgap fabrication facility prior to full commercialization of quantum computing. Operating a quantum computer will likely be roughly comparable with operating GPU racks in the sense that quantum computers have sophisticated cooling requirements and can also be energy-intensive to operate. For example, today’s superconducting quantum computers require roughly 20 kW to operate, a majority of which is consumed by the dilution refrigerator and microwave electronics.

In this paper, we have outlined how a consortium across semiconductor research efforts could lower research costs: 1) the advancement to nanoscale transistors requires atomistic control of fabrication parameters, which in turn could allow suppression of two-level system defects; 2) removing memory bottlenecks for larger AI chips in turn enables cryogenic wafer-scale integration; 3) reducing AI chip energy consumption by means of cryoCMOS can be used for scalable RF control; 4) scaling RF control for technologies like phase-array microwave receivers can apply to scalable quantum control; and 5) integrating quantum computers with heterogeneous high-performance computation powered by various classical hardware accelerators including GPU, FPGA, and PPU (specialized ASICs and custom-designed hardware for probabilistic computing) can provide an improvement by orders of magnitude in speed and energy efficiency. The benefits of integrating existing semiconductor research into quantum computing can be mutual.

A natural follow-up question for utility-scale computing is the estimated timeline for development. This will be addressed next.

B. Exponential scaling progress: A mirage or reality?

Since the quantum supremacy milestone [4], there has been an expectation that scaling the number of qubits will follow a Moore’s law (exponential) growth over time from ~ 50 qubits to a million qubits at the end of this decade. This goal would mark the arrival of a practical fault-tolerant quantum computer. We chart this progress in Figure 42, starting from 2014 when a repetition code experiment was performed at UCSB (nine qubits) [202], through the Google quantum supremacy on random circuits [4] (53 qubits), on to the most recent surface-code error-correction experiment (105 qubits) [26]. Care must

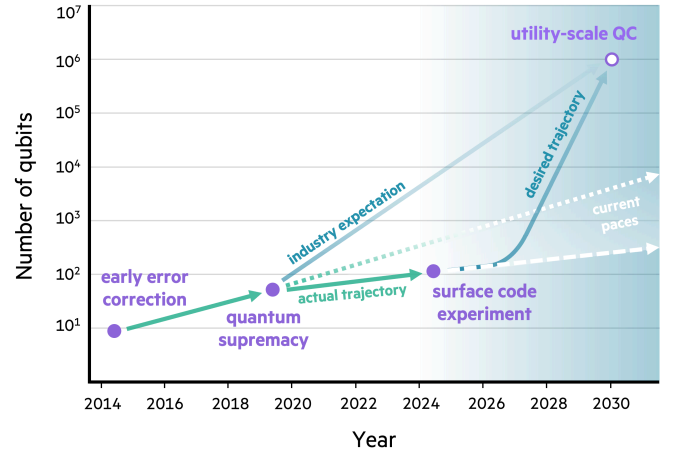


Figure 42: Schematic plot of the number of qubits from three experiments at UCSB and Google over time, which describes a Moore’s law growth in the number of qubits. After the quantum supremacy experiment in 2019 [4], a target of one million qubits at the end of the decade was projected by industry leaders. However, the current pace of hardware progress suggests that goal might be postponed by several decades, assuming an optimistic scenario that none of the scaling challenges mentioned here slows or halts the progress. To arrive at utility-scale quantum computers in the 2030-2035 time frame, we need a major increase in the rate of progress over the next five years. Our thesis is that new fabrication and systems design as well as full-stack HPC integration are required to tackle this challenge. For estimates of future scaling, we suggest using the number of qubits that can be entangled in practice (which assumes sufficient qubit connectivity with multiqubit gates that are fast and accurate enough)[18].

be taken in plotting only the qubit quantity since this ignores other important qubit metrics such as quality, speed, and connectivity. Here, these data points represent qubit systems with some degree of consistency in these four key metrics, thus it is a fairly reasonable plot.

The trajectory of these three data points appears to follow an exponential improvement, but at a much lower slope than is needed to reach one million qubits by 2030. Larger numbers of qubits have been reported for superconducting quantum processors, but even for these more optimistic characterizations of functional qubits, the scaling falls short of the original industry expectation. Will it be possible to greatly increase the slope over the next five years, corresponding to a double exponential growth? We think this is unlikely because the added challenges for scaling beyond 1000 qubits—detailed in Section IB—could hinder even staying on the current growth path.

As we have outlined in this position paper, one way to increase the rate of exponential growth is to identify all major technical challenges that are blocking progress and devise mitigation strategies. These include taking a radically different approach to qubit fabrication and developing a full-stack system integration with heterogeneous high-performance computing infrastructures.

We have also studied the detailed trade-off of physical and computational resources for scalable error-corrected quantum computers based on resource estimates of classically hard electronic structure calculations leveraging realistic performance characteristics for superconducting

qubits. We showed for quantum simulations of FeMoco with chemical accuracy $\epsilon = 1.0$ mHa, one requires hundreds of millions of physical qubits and a minimum of 2.5 years to run with state-of-the-art hardware quality (baseline hardware). The runtime can be reduced to approximately half a year and the qubit count to tens of millions if the hardware quality is significantly improved towards desired hardware. Improving the hardware quality from baseline to desired values results in a reduced runtime and qubit count of approximately a factor of five. Our sensitivity analysis shows that improvements in the gate-control errors yield the most significant impact, whereas improvements in SPAM errors and coherence enhancements are significantly less effective for achieving better performance in error suppression. These findings suggest that quantum gate fidelity improvements are much more important than SPAM or idling qubit error rates for scaling logical performance. Furthermore, we demonstrate the robustness of lattice surgeries spread among separate capacitively coupled QPU wafers and even separate QPUs distributed among multiple DRs. We conclude that distributed surface code architectures across multiple DRs can tolerate two-qubit errors on the order of 1% arising from noisy optical interconnects between the DRs.

We have attempted to provide a comprehensive list of all known technical challenges for scaling quantum processors, the goal being to illuminate obstacles that have been overlooked or addressed piecemeal in prior research [7–9]. Contemporary quantum computing platforms have primarily concentrated on technical hurdles at the hundred-qubit level, constrained by the quality of individual qubits. By anticipating technical scaling chal-

lenges from a thousand to a million qubits, our study outlines a holistic system that could provide practical quantum advantage. Not all of these challenges have been addressed in detail in this paper. Our hope is that, by publicizing the obstacles to scaling a quantum supercomputer, we can stimulate the discovery of innovative solutions in both academia and industry. Future revisions of this approach are expected with new discoveries and collaborators.

Acknowledgments. The authors from HPE are supported by the Defense Advanced Research Projects Agency (DARPA) under Air Force Research Laboratory (AFRL) contract no. FA8650-23-3-7313. We would like to thank Eleanor Rieffel for useful discussions and review of the manuscript. Authors from NASA/USRA acknowledge NASA Academic Mission Service contract No. NNA16BD14C. The authors from 1QBit thank our editor, Marko Bucyk, for editorial review of the manuscript. We are grateful to Alexandre Fleury, Einar Gabassov, Mia Kramer, Huy Anh Nguyen, Kevin Nguyen, Katie Olfert, Valentin Senicourt, Yumeng Wang, Chan Woo Yang, and Xiangyi Zhang for useful discussions and support. The authors acknowledge the financial support of Pacific Economic Development Canada (PacifiCan) under project number PC0008525. G. A. M. is grateful for the support of Mitacs. P. R. acknowledges the financial support of Mike and Ophelia Lazaridis, Innovation, Science and Economic Development Canada (ISED), and the Perimeter Institute for Theoretical Physics. Research at the Perimeter Institute is supported in part by the Government of Canada through ISED and by the Province of Ontario through the Ministry of Colleges and Universities.

-
- [1] P. Shor, in *Proceedings 35th Annual Symposium on Foundations of Computer Science* (1994) pp. 124–134.
 - [2] P. W. Shor, *Phys. Rev. A* **52**, R2493 (1995).
 - [3] D. Gottesman, “Stabilizer codes and quantum error correction,” (1997), arXiv:quant-ph/9705052 [quant-ph].
 - [4] F. Arute et al., *Nature* **574**, 505 (2019).
 - [5] “Quantum advantage,” (2023).
 - [6] J. Altepeter, “Underexplored systems for utility-scale quantum computing,” (2022).
 - [7] Y. e. a. Alexeev, *Future Generation Computer Systems* **160**, 666 (2024).
 - [8] S. Bravyi, O. Dial, J. M. Gambetta, D. Gil, and Z. Nazario, *Journal of Applied Physics* **132**, 160902 (2022).
 - [9] T. S. Humble, A. McCaskey, D. I. Lyakh, M. Gowardhankar, A. Frisch, and T. Monz, *IEEE Micro* **41**, 15 (2021).
 - [10] J. Preskill, *Quantum* **2**, 79 (2018).
 - [11] W. K. Wootters and W. H. Zurek, *Nature* **299**, 802 (1982).
 - [12] D. Aharonov and M. Ben-Or, in *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, STOC ’97 (Association for Computing Machinery, New York, NY, USA, 1997) p. 176–188.
 - [13] E. Knill, R. Laflamme, and W. H. Zurek, *Science* **279**, 342 (1998), <https://www.science.org/doi/pdf/10.1126/science.279.5349.342>.
 - [14] A. Kitaev, *Annals of Physics* **303**, 2–30 (2003).
 - [15] L. Ding et al., *Phys. Rev. X* **102**, 110502 (2023).
 - [16] M. McEwen et al., “Resisting high-energy impact events through gap engineering in superconducting qubit arrays,” (2024), arXiv:2402.15644 [quant-ph].
 - [17] “The ibm quantum heavy hex lattice,” (2021).
 - [18] S. Cao et al., *Nature* **619**, 738–742 (2023).
 - [19] A. Maksymov, J. Nguyen, Y. Nam, and I. L. Markov, (2023), arXiv:2023.07233 [quant-ph].
 - [20] J. P. Bonilla Ataides, D. K. Tuckett, S. D. Bartlett, S. T. Flammia, and B. J. Brown, *Nature Communications* **12** (2021), 10.1038/s41467-021-22274-1.
 - [21] P.-K. Tsai, Y. Wu, and S. Puri, *Phys. Rev. X* **14**, 031003 (2024).
 - [22] I. L. Markov, *Nature* **512**, 147–154 (2014).
 - [23] A. Y. Kitaev, *Annals of physics* **303**, 2 (2003).
 - [24] R. Acharya et al., *Nature* **614**, 676 (2023).
 - [25] J. M. Martinis, arXiv preprint arXiv:2012.06137 (2020).
 - [26] R. Acharya, L. Aghababaie-Beni, I. Aleiner, T. I. An-

- dersen, M. Ansmann, F. Arute, K. Arya, A. Asfaw, N. Astrakhantsev, J. Atalaya, *et al.*, arXiv preprint arXiv:2408.13687 (2024).
- [27] L. Caune *et al.*, “Demonstrating real-time and low-latency quantum error correction with superconducting qubits,” (2024), arXiv:2410.05202 [quant-ph].
- [28] D. Litinski, *Quantum* **3**, 128 (2019).
- [29] T. Peng, A. W. Harrow, M. Ozols, and X. Wu, *Physical review letters* **125**, 150504 (2020).
- [30] C. Piveteau, D. Sutter, and S. Woerner, *npj Quantum Information* **8**, 12 (2022).
- [31] C. Piveteau and D. Sutter, *IEEE Transactions on Information Theory* (2023).
- [32] K. C. Smith, E. Crane, N. Wiebe, and S. Girvin, *PRX Quantum* **4**, 020315 (2023).
- [33] K. C. Smith, A. Khan, B. K. Clark, S. Girvin, and T.-C. Wei, *PRX Quantum* **5**, 030344 (2024).
- [34] M. Foss-Feig, A. Tikku, T.-C. Lu, K. Mayer, M. Iqbal, T. M. Gatterman, J. A. Gerber, K. Gilmore, D. Gresh, A. Hankin, *et al.*, arXiv preprint arXiv:2302.03029 (2023).
- [35] M. Reiher, N. Wiebe, K. M. Svore, D. Wecker, and M. Troyer, *Proceedings of the National Academy of Sciences* **114**, 7555 (2017).
- [36] Z. Li, J. Li, N. S. Dattani, C. J. Umrigar, and G. K.-L. Chan, *The Journal of Chemical Physics* **150** (2019), 10.1063/1.5063376.
- [37] N. P. Breuckmann and J. N. Eberhardt, *PRX Quantum* **2** (2021), 10.1103/prxquantum.2.040101.
- [38] D. Gottesman, “Opportunities and challenges in fault-tolerant quantum computation,” (2022), 2210.15844.
- [39] L. Lavagno, G. E. Martin, L. Scheffer, and I. L. Markov, eds., *Electronic Design Automation for Integrated Circuits Handbook: EDA for IC system design, verification, and testing* (CRC/Taylor and Francis, 2016).
- [40] S. Yamashita and I. L. Markov, *Quantum Inf. Comput.* **10**, 721 (2010).
- [41] A. Maksymov, J. Nguyen, V. Chaplin, Y. S. Nam, and I. L. Markov, in *Proc. HPCA* (2020) pp. 387–399.
- [42] K. Romanik, *Theoretical Computer Science* **188**, 79 (1997).
- [43] S. A. Metwalli and R. V. Meter, (2023), arXiv:2311.18202 [quant-ph].
- [44] B. S. Lee, B. Kim, A. P. Freitas, A. Mohanty, Y. Zhu, G. R. Bhatt, J. Hone, and M. Lipson, *Nanophotonics* **10**, 99 (2020).
- [45] A. Youssefi, I. Shomroni, Y. J. Joshi, N. R. Bernier, A. Lukashchuk, P. Urich, L. Qiu, and T. J. Kippenberg, *Nature Electronics* **4**, 326 (2021).
- [46] F. Lecocq, F. Quinlan, K. Cicak, J. Aumentado, S. Didams, and J. Teufel, *Nature* **591**, 575 (2021).
- [47] D. Awschalom, K. K. Berggren, H. Bernien, S. Bhawe, L. D. Carr, P. Davids, S. E. Economou, D. Englund, A. Faraon, M. Fejer, *et al.*, *Prx Quantum* **2**, 017002 (2021).
- [48] R. Delaney, M. Urmey, S. Mittal, B. Brubaker, J. Kindem, P. Burns, C. Regal, and K. Lehnert, *Nature* **606**, 489 (2022).
- [49] M. E. Beverland, P. Murali, M. Troyer, K. M. Svore, T. Hoefler, V. Kliuchnikov, G. H. Low, M. Soeken, A. Sundaram, and A. Vassillo, “Assessing requirements to scale to practical quantum advantage,” (2022), arXiv:2211.07629 [quant-ph].
- [50] A. Silva, A. Scherer, Z. Webb, A. Khalid, B. Kulchyt-sky, M. Kramer, K. Nguyen, X. Kong, G. A. Dagnew, Y. Wang, H. A. Nguyen, K. Olfert, and P. Ronagh, “Optimizing multi-level magic state factories for fault-tolerant quantum architectures,” (2024), arXiv:2411.04270 [quant-ph].
- [51] A. Silva, X. Zhang, Z. Webb, M. Kramer, C. W. Yang, J. L. Xiao Liu, K.-W. Chen, A. Scherer, and P. Ronagh, in *Proceedings of the 19th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2024)* (Leibniz International Proceedings in Informatics (LIPIcs), 2024).
- [52] N. Verma, H. Jia, H. Valavi, Y. Tang, M. Ozatay, L.-Y. Chen, B. Zhang, and P. Deaville, *IEEE Solid-State Circuits Magazine* **11**, 43 (2019).
- [53] V. Giovannetti, S. Lloyd, and L. Maccone, *Physical Review Letters* **100** (2008), 10.1103/physrevlett.100.160501.
- [54] O. D. Matteo, V. Gheorghiu, and M. Mosca, *IEEE Transactions on Quantum Engineering* **1**, 1–13 (2020).
- [55] A. W. Harrow, A. Hassidim, and S. Lloyd, *Phys. Rev. Lett.* **103**, 150502 (2009).
- [56] A. M. Childs, R. Kothari, and R. D. Somma, *SIAM Journal on Computing* **46**, 1920–1950 (2017).
- [57] D. Horn and A. Gottlieb, *Phys. Rev. Lett.* **88**, 018702 (2001).
- [58] S. Lloyd, M. Mohseni, and P. Rebentrost, *Nature Physics* **10**, 631–633 (2014).
- [59] P. Rebentrost, M. Mohseni, and S. Lloyd, *Physical Review Letters* **113** (2014), 10.1103/physrevlett.113.130503.
- [60] H.-Y. Huang, M. Broughton, J. Cotler, S. Chen, J. Li, M. Mohseni, H. Neven, R. Babbush, R. Kueng, J. Preskill, and J. R. McClean, *Science* **376**, 1182 (2022), <https://www.science.org/doi/pdf/10.1126/science.abn7293>.
- [61] M. Larocca, S. Thanasilp, S. Wang, K. Sharma, J. Biamonte, P. J. Coles, L. Cincio, J. R. McClean, Z. Holmes, and M. Cerezo, arXiv preprint arXiv:2405.00781 (2024).
- [62] T. Häner, M. Roetteler, and K. M. Svore, “Optimizing quantum circuits for arithmetic,” (2018), arXiv:1805.12445 [quant-ph].
- [63] T. Häner, M. Soeken, M. Roetteler, and K. M. Svore, “Quantum circuits for floating-point arithmetic,” (2018), arXiv:1807.02023 [quant-ph].
- [64] R. Babbush, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, A. Paler, A. Fowler, and H. Neven, *Phys. Rev. X* **8**, 041015 (2018).
- [65] P. Høyer, J. Neerbek, and Y. Shi, “Quantum complexities of ordered searching, sorting, and element distinctness,” in *Automata, Languages and Programming* (Springer Berlin Heidelberg, 2001) p. 346–357.
- [66] D. P. Rodgers, *SIGARCH Comput. Archit. News* **13**, 225–231 (1985).
- [67] M. Szegedy, in *45th Annual IEEE symposium on foundations of computer science* (IEEE, 2004) pp. 32–41.
- [68] R. Babbush, J. R. McClean, M. Newman, C. Gidney, S. Boixo, and H. Neven, *PRX Quantum* **2**, 010103 (2021).
- [69] E. Farhi, J. Goldstone, and S. Gutmann, arXiv preprint arXiv:1411.4028 (2014).
- [70] K. Blekos, D. Brand, A. Ceschini, C.-H. Chou, R.-H. Li, K. Pandya, and A. Summer, *Physics Reports* **1068**, 1 (2024).

- [71] R. Shaydulin, C. Li, S. Chakrabarti, M. DeCross, D. Herman, N. Kumar, J. Larson, D. Lykov, P. Minssen, Y. Sun, Y. Alexeev, J. M. Dreiling, J. P. Gaebler, T. M. Gatterman, J. A. Gerber, K. Gilmore, D. Gresh, N. Hewitt, C. V. Horst, S. Hu, J. Johansen, M. Matheny, T. Mengle, M. Mills, S. A. Moses, B. Neyenhuis, P. Siegfried, R. Yalovetzky, and M. Pistoia, *Science Advances* **10**, eadm6761 (2024), <https://www.science.org/doi/pdf/10.1126/sciadv.adm6761>.
- [72] S. Boulebnane and A. Montanaro, *PRX Quantum* **5**, 030348 (2024).
- [73] B. Augustino, M. Cain, E. Farhi, S. Gupta, S. Gutmann, D. Ranard, E. Tang, and K. Van Kirk, arXiv preprint arXiv:2410.03015 (2024).
- [74] A. Montanaro and L. Zhou, (2024), arXiv:2411.04979v1, 2411.04979.
- [75] D. W. Berry, Y. Tong, T. Khatrar, A. White, T. I. Kim, S. Boixo, L. Lin, S. Lee, G. K.-L. Chan, R. Babbush, and N. C. Rubin, “Rapid initial state preparation for the quantum simulation of strongly correlated molecules,” (2024), arXiv:2409.11748 [quant-ph].
- [76] C. Liu *et al.*, *Phys. Rev. Lett.* **132**, 017001 (2024).
- [77] J. Rochman, M. Guibord, H. Yan, L. Du, Z. Yang, R. Li, M. Zhu, N. Patibandla, and R. J. Visser, in *31st International Display Workshops (IDW '24)* (2024).
- [78] Cerebras, “Wafer scale engine,” (2023).
- [79] NVidia, “Nvidia blackwell platform arrives to power a new era of computing,” (2024).
- [80] A. Materials, “Heterogeneous integration,” (2024).
- [81] A. Opremcak *et al.*, *Phys. Rev. X* **102**, 011027 (2020).
- [82] M. Mirhosseini, A. Sipahigil, M. Kalaei, and O. Painter, *Nature* **588** (2023).
- [83] L. Ella, L. Leandro, O. Wertheim, Y. Romach, L. Schlipf, R. Szmuk, Y. Knol, N. Ofek, I. Sivan, and Y. Cohen, “Quantum-classical processing and benchmarking at the pulse-level,” (2023), arXiv:2303.03816 [quant-ph].
- [84] P. W. Shor, *Physical review A* **52**, R2493 (1995).
- [85] P. W. Shor, in *Proceedings of 37th conference on foundations of computer science* (IEEE, 1996) pp. 56–65.
- [86] J. Preskill, “Fault-tolerant quantum computation,” (1997), arXiv:quant-ph/9712048 [quant-ph].
- [87] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, *Journal of Mathematical Physics* **43**, 4452–4505 (2002).
- [88] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, *Physical Review A* **86** (2012), 10.1103/physreva.86.032324.
- [89] H. Bombin and M. A. Martin-Delgado, *Phys. Rev. Lett.* **97**, 180501 (2006).
- [90] A. J. Landahl, J. T. Anderson, and P. R. Rice, “Fault-tolerant quantum computing with color codes,” (2011), arXiv:1108.5738 [quant-ph].
- [91] B. M. Terhal, *Reviews of Modern Physics* **87**, 307 (2015).
- [92] C. Wang, J. Harrington, and J. Preskill, *Annals of Physics* **303**, 31 (2003).
- [93] H. Bombin and M. A. Martin-Delgado, *Phys. Rev. A* **76**, 012305 (2007).
- [94] D. Horsman, A. G. Fowler, S. Devitt, and R. V. Meter, *New Journal of Physics* **14**, 123011 (2012).
- [95] B. Eastin and E. Knill, *Phys. Rev. Lett.* **102**, 110502 (2009).
- [96] S. Singh, A. S. Darmawan, B. J. Brown, and S. Puri, *Phys. Rev. A* **105**, 052410 (2022).
- [97] C. Gidney, arXiv preprint arXiv:2302.12292 (2023).
- [98] C. Gidney, N. Shutty, and C. Jones, arXiv preprint arXiv:2409.17595 (2024).
- [99] 1QB Information Technologies (1QBit), “TopQAD: Topological Quantum Architecture Design [Software Documentation],” (2024).
- [100] S. Bravyi and A. Kitaev, *Phys. Rev. A* **71**, 022316 (2005).
- [101] Z. Chen *et al.*, *Nature* **595**, 383–387 (2021).
- [102] C. Gidney, *Quantum* **5**, 497 (2021).
- [103] O. Higgott, *ACM Transactions on Quantum Computing* **3**, 1 (2022).
- [104] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, *Physical Review A—Atomic, Molecular, and Optical Physics* **86**, 032324 (2012).
- [105] I. Quantum, Accessed: 2024-10-09.
- [106] T. K. Ho, in *Proceedings of 3rd international conference on document analysis and recognition*, Vol. 1 (IEEE, 1995) pp. 278–282.
- [107] C. Chamberland and E. T. Campbell, *PRX Quantum* **3** (2022), 10.1103/prxquantum.3.010331.
- [108] M. A. Nielsen, *Physics Letters A* **303**, 249–252 (2002).
- [109] C. Ryan-Anderson, N. C. Brown, C. H. Baldwin, J. M. Dreiling, C. Foltz, J. P. Gaebler, T. M. Gatterman, N. Hewitt, C. Holliman, C. V. Horst, J. Johansen, D. Lucchetti, T. Mengle, M. Matheny, Y. Matsuoka, K. Mayer, M. Mills, S. A. Moses, B. Neyenhuis, J. Pino, P. Siegfried, R. P. Stutz, J. Walker, and D. Hayes, “High-fidelity and fault-tolerant teleportation of a logical qubit using transversal gates and lattice surgery on a trapped-ion quantum computer,” (2024), arXiv:2404.16728 [quant-ph].
- [110] C. Gidney, *Quantum* **8**, 1310 (2024).
- [111] Y. Kurman, L. Ella, R. Szmuk, O. Wertheim, B. Dorschner, S. Stanwyck, and Y. Cohen, “Control requirements and benchmarks for quantum error correction,” (2023), arXiv:2311.07121 [quant-ph].
- [112] Microsoft, “Azure Quantum Resource Estimator,” (2022).
- [113] A. Szabo and N. S. Ostlund, *Modern quantum chemistry: introduction to advanced electronic structure theory* (Courier Corporation, 1996).
- [114] T. D. Crawford, E. Kraka, J. F. Stanton, and D. Cremer, *The Journal of Chemical Physics* **114**, 10638 (2001).
- [115] E. Kraka and D. Cremer, *Journal of the American Chemical Society* **122**, 8245 (2000).
- [116] M. Motta, E. Ye, J. R. McClean, Z. Li, A. J. Minnich, R. Babbush, and G. K.-L. Chan, *npj Quantum Information* **7** (2021), 10.1038/s41534-021-00416-z.
- [117] J. Lee, D. W. Berry, C. Gidney, W. J. Huggins, J. R. McClean, N. Wiebe, and R. Babbush, *PRX Quantum* **2**, 030305 (2021).
- [118] M. Otten, B. Kang, D. Fedorov, J.-H. Lee, A. Benali, S. Habib, S. K. Gray, and Y. Alexeev, *Frontiers in Quantum Science and Technology* **2**, 1232624 (2023).
- [119] A. Y. Kitaev, “Quantum measurements and the abelian stabilizer problem,” (1995), arXiv:quant-ph/9511026 [quant-ph].
- [120] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2011).
- [121] P. Jordan and E. P. Wigner, “Über das paulische äquivalenzverbot,” in *The Collected Works of Eugene Paul*

- Wigner: Part A: The Scientific Papers*, edited by A. S. Wightman (Springer Berlin Heidelberg, Berlin, Heidelberg, 1993) pp. 109–129.
- [122] S. B. Bravyi and A. Y. Kitaev, *Annals of Physics* **298**, 210 (2002).
 - [123] V. Senicourt, J. Brown, A. Fleury, R. Day, E. Lloyd, M. P. Coons, K. Bieniasz, L. Huntington, A. J. Garza, S. Matsuura, R. Plesch, T. Yamazaki, and A. Zaribafian, (2022), 10.48550/arXiv.2206.12424, arXiv:2206.12424.
 - [124] J. Lee, D. W. Berry, C. Gidney, W. J. Huggins, J. R. McClean, N. Wiebe, and R. Babbush, “Data and code repository for “Even more efficient quantum computations of chemistry through tensor hypercontraction”,” (2020).
 - [125] G. H. Low and I. L. Chuang, *Quantum* **3**, 163 (2019).
 - [126] D. W. Berry, C. Gidney, M. Motta, J. R. McClean, and R. Babbush, *Quantum* **3**, 208 (2019).
 - [127] V. von Burg, G. H. Low, T. Häner, D. S. Steiger, M. Reiher, M. Roetteler, and M. Troyer, *Phys. Rev. Res.* **3**, 033055 (2021).
 - [128] J. J. Goings, A. White, J. Lee, C. S. Tautermann, M. Degroote, C. Gidney, T. Shiozaki, R. Babbush, and N. C. Rubin, *Proceedings of the National Academy of Sciences* **119**, e2203533119 (2022), <https://www.pnas.org/doi/pdf/10.1073/pnas.2203533119>.
 - [129] M. Otten, T. W. Watts, S. D. Johnson, R. Sundareswara, Z. Wang, T. S. Hardikar, K. Heitritter, J. Brown, K. Setia, and A. Holmes, arXiv preprint arXiv:2406.18744 (2024).
 - [130] N. Bellonzi, A. Kunitsa, J. T. Cantin, J. A. Campos-Gonzalez-Angulo, M. D. Radin, Y. Zhou, P. D. Johnson, L. A. Martínez-Martínez, M. R. Jangrouei, A. S. Brahmachari, *et al.*, arXiv preprint arXiv:2406.06335 (2024).
 - [131] T. W. Watts, M. Otten, J. T. Necaie, N. Nguyen, B. Link, K. S. Williams, Y. R. Sanders, S. J. Elman, M. Kieferova, M. J. Bremner, *et al.*, arXiv preprint arXiv:2408.13244 (2024).
 - [132] N. Nguyen, T. W. Watts, B. Link, K. S. Williams, Y. R. Sanders, S. J. Elman, M. Kieferova, M. J. Bremner, K. J. Morrell, J. Elenewski, *et al.*, arXiv preprint arXiv:2406.18759 (2024).
 - [133] D. W. Berry, M. Kieferová, A. Scherer, Y. R. Sanders, G. H. Low, N. Wiebe, C. Gidney, and R. Babbush, *npj Quantum Information* **4**, 1 (2018).
 - [134] D. Poulin, A. Kitaev, D. S. Steiger, M. B. Hastings, and M. Troyer, *Phys. Rev. Lett.* **121**, 010501 (2018).
 - [135] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan, *Rev. Mod. Phys.* **92**, 015003 (2020).
 - [136] gridsynth: An open-source software package to compute approximations of arbitrary-angle z-rotations over the Clifford+T gate set,.
 - [137] P. Selinger, *Quantum Info. Comput.* **15**, 159–180 (2015).
 - [138] N. J. Ross and P. Selinger, *Quantum Info. Comput.* **16**, 901–953 (2016).
 - [139] S. R. White, *Phys. Rev. Lett.* **69**, 2863 (1992).
 - [140] H. Gao, S. Imamura, A. Kasagi, and E. Yoshida, *Journal of Chemical Theory and Computation* **20**, 1185 (2024).
 - [141] Nvidia, “Nvidia/cuda-quantum,” .
 - [142] IBM, “Introduction to qiskit,” .
 - [143] Google, “Cirq: An open source framework for programming quantum computers,” .
 - [144] Xanadu, “PennyLane,” .
 - [145] Classiq Technologies Ltd., “Classiq platform,” .
 - [146] NVIDIA, “Cuda-q,” .
 - [147] A. Asadi, A. Dusko, C.-Y. Park, V. Michaud-Rioux, I. Schoch, S. Shu, T. Vincent, and L. J. O’Riordan, arXiv:2403.02512 (2024).
 - [148] Slurm, “Slurm workload manager,” .
 - [149] OpenPBS, “Open pbs open source project,” .
 - [150] H. Jose Morrell Jr, A. Zaman, and H. Y. Wong, arXiv:2108.09004 (2021).
 - [151] A. Esposito and T. Danzig, in *2024 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)* (IEEE, 2024) pp. 1088–1094.
 - [152] A. W. Cross, L. S. Bishop, J. A. Smolin, and J. M. Gambetta, arXiv:1707.03429 (2017).
 - [153] R. S. Smit, M. J. Curtis, and W. J. Zeng, arXiv:1608.03355v2 (2017).
 - [154] A. Litteken, Y.-C. Fan, D. Singh, M. Martonosi, and F. T. Chong, *Quantum Science and Technology* **5**, 034013 (2020).
 - [155] F. T. Chong, D. Franklin, and M. Martonosi, *Nature* **549**, 180–187 (2017).
 - [156] Microsoft, “Quantum intermediate representation (qir) specification,” .
 - [157] S. R. White, *Physical review b* **48**, 10345 (1993).
 - [158] H.-D. Meyer, U. Manthe, and L. S. Cederbaum, *Chemical Physics Letters* **165**, 73 (1990).
 - [159] U. Manthe, *The Journal of chemical physics* **128** (2008).
 - [160] K. Temme, S. Bravyi, and J. M. Gambetta, *Phys. Rev. Lett.* **119**, 180509 (2017).
 - [161] W. Tang, T. Tomesh, M. Suchara, J. Larson, and M. Martonosi, in *Proceedings of the 26th ACM international conference on architectural support for programming languages and operating systems* (2021) pp. 473–486.
 - [162] W. Tang and M. Martonosi, arXiv preprint arXiv:2207.00933 (2022).
 - [163] S. Basu, A. Das, A. Saha, A. Chakrabarti, and S. Surkolay, *Journal of Systems and Software* **214**, 112085 (2024).
 - [164] A. Skolik, J. R. McClean, M. Mohseni, P. Van Der Smagt, and M. Leib, *Quantum Machine Intelligence* **3**, 1 (2021).
 - [165] J. Carolan, M. Mohseni, J. P. Olson, M. Prabhu, C. Chen, D. Bunandar, M. Y. Niu, N. C. Harris, F. N. Wong, M. Hochberg, *et al.*, *Nature Physics* **16**, 322 (2020).
 - [166] M. Broughton, G. Verdon, T. McCourt, A. J. Martinez, J. H. Yoo, S. V. Isakov, P. Massey, R. Halavati, M. Y. Niu, A. Zlokapa, E. Peters, O. Lockwood, A. Skolik, S. Jerbi, V. Dunjko, M. Leib, M. Streif, D. Von Dollen, H. Chen, S. Cao, R. Wiersema, H.-Y. Huang, J. R. McClean, R. Babbush, S. Boixo, D. Bacon, A. K. Ho, H. Neven, and M. Mohseni, (2020), arXiv:2003.02989 [quant-ph].
 - [167] M. Heyl, *Reports on Progress in Physics* **81**, 054001 (2018).
 - [168] H. F. Trotter, *Proceedings of the American Mathematical Society* **10**, 545 (1959).
 - [169] M. Suzuki, *Communications in Mathematical Physics* **51**, 183 (1976).
 - [170] J. Dborin, V. Wimalaweera, F. Barratt, E. Ostby, T. E. O’Brien, and A. G. Green, *Nature Communications* **13**, 5977 (2022).
 - [171] J. Zhang, G. Pagano, P. W. Hess, A. Kyprianidis,

- P. Becker, H. Kaplan, A. V. Gorshkov, Z.-X. Gong, and C. Monroe, *Nature* **551**, 601 (2017).
- [172] S.-H. Lin, R. Dilip, A. G. Green, A. Smith, and F. Pollmann, *PRX Quantum* **2**, 010342 (2021).
- [173] K. Y. Camsari *et al.*, *Physical Review X* **7**, 031014 (2017).
- [174] K. Y. Camsari, S. Salahuddin, and S. Datta, *IEEE Electron Device Letters* **38**, 1767 (2017).
- [175] N. A. Aadit, A. Grimaldi, M. Carpentieri, L. Theogarajan, J. M. Martinis, G. Finocchio, and K. Y. Camsari, *Nature Electronics* **5**, 460 (2022).
- [176] B. Sutton, R. Faria, L. A. Ghantasala, R. Jaiswal, K. Y. Camsari, and S. Datta, *IEEE Access* **8**, 157238 (2020).
- [177] S. Niazi, S. Chowdhury, N. A. Aadit, M. Mohseni, Y. Qin, and K. Y. Camsari, *Nature Electronics*, 1 (2024).
- [178] N. S. Singh, K. Kobayashi, Q. Cao, K. Selcuk, T. Hu, S. Niazi, N. A. Aadit, S. Kanai, H. Ohno, S. Fukami, *et al.*, *Nature Communications* **15**, 2685 (2024).
- [179] S. Nikhar, S. Kannan, N. A. Aadit, S. Chowdhury, and K. Y. Camsari, *Nature Communications* **15**, 8977 (2024).
- [180] W. Whitehead, Z. Nelson, K. Y. Camsari, and L. Theogarajan, *Nature Electronics* **6**, 1009 (2023).
- [181] G. Pedretti, F. Böhm, M. Hizzani, T. Bhattacharya, P. Bruel, J. Moon, S. Serebryakov, D. Strukov, J. Strachan, J. Ignowski, *et al.*, in *2023 International Electron Devices Meeting (IEDM)* (IEEE, 2023) pp. 1–4.
- [182] T. Bhattacharya, G. H. Hutchinson, G. Pedretti, X. Sheng, J. Ignowski, T. Van Vaerenbergh, R. Beau-soleil, J. P. Strachan, and D. B. Strukov, *Nature Communications* **15**, 8211 (2024).
- [183] D. Dobrynin, A. Renaudineau, M. Hizzani, D. Strukov, M. Mohseni, and J. P. Strachan, *Physical Review E* **110**, 045308 (2024).
- [184] W. A. Borders, K. Y. Camsari, *et al.*, *Nature* (2019).
- [185] K. Lee, J. Bak, Y. Kim, C. Kim, A. Antonyan, D. Chang, S. Hwang, G. Lee, N. Ji, W. Kim, *et al.*, in *2019 IEEE International Electron Devices Meeting (IEDM)* (IEEE, 2019) pp. 2–2.
- [186] J. Kaiser and S. Datta, *Applied Physics Letters* **119**, 150503 (2021).
- [187] R. M. Radway, A. Bartolo, P. C. Jolly, Z. F. Khan, B. Q. Le, P. Tandon, T. F. Wu, Y. Xin, E. Vianello, P. Vivet, *et al.*, *Nature Electronics* **4**, 71 (2021).
- [188] R. Liao, S. Kornblith, M. Ren, D. J. Fleet, and G. Hinton, *arXiv preprint arXiv:2210.10318* (2022).
- [189] R. Bittner, E. Ruf, and A. Forin, *Cluster Computing* **17**, 339 (2014).
- [190] K. Lim, J. Chang, T. Mudge, P. Ranganathan, S. K. Reinhardt, and T. F. Wenisch, *ACM SIGARCH computer architecture news* **37**, 267 (2009).
- [191] W. Jiang, M. Zeller, R. Waleffe, T. Hoeffer, and G. Alonso, *arXiv preprint arXiv:2310.09949* (2023).
- [192] K. Cao, A. Gajjar, L. Gerstman, K. Wu, S. R. Chalamalasetti, A. Dhakal, G. Pedretti, P. Prakash, W.-m. Hwu, D. Chen, and D. Milojicic, in *2024 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC)* (2024).
- [193] T. Kadowaki and H. Nishimori, *Physical Review E* **58**, 5355 (1998).
- [194] T. Albash and D. A. Lidar, *Reviews of Modern Physics* **90**, 015002 (2018).
- [195] V. S. Denchev, M. Mohseni, and H. Neven, “Quantum assisted optimization,” (2022), *uS Patent* 11,449,760.
- [196] J. King, M. Mohseni, W. Bernoudy, A. Fréchet, H. Sadeghi, S. V. Isakov, H. Neven, and M. H. Amin, (2019), *arXiv:1907.00707* [quant-ph].
- [197] M. Mohseni, D. Eppens, J. Strumpfer, R. Marino, V. Denchev, A. K. Ho, S. V. Isakov, S. Boixo, F. Ricci-Tersenghi, and H. Neven, *arXiv preprint arXiv:2111.13628* (2021).
- [198] K. Kechedzhi, V. Smelyanskiy, J. R. McClean, V. S. Denchev, M. Mohseni, S. Isakov, S. Boixo, B. Altshuler, and H. Neven, *arXiv:1807.04792* (2018).
- [199] M. Mohseni and H. Neven, “Chips including classical and quantum computing processors,” (2020), *uS Patent* 10,671,559.
- [200] M. Mohseni, M. M. Rams, S. V. Isakov, D. Eppens, S. Pielawa, J. Strumpfer, S. Boixo, and H. Neven, *Phys. Rev. E* **108** (2023).
- [201] A. Zucca, H. Sadeghi, M. Mohseni, and M. H. Amin, *arXiv:2110.10196* (2021).
- [202] J. Kelly, R. Barends, A. G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, J. Y. Mutus, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, I.-C. Hoi, C. Neill, P. J. J. O’Malley, C. Quintana, P. Roushan, A. Vainsencher, J. Wenner, A. N. Cleland, and J. M. Martinis, *Nature* **519** (2015).
- [203] A. Y. Kitaev, “Quantum measurements and the abelian stabilizer problem,” (1995), *arXiv:quant-ph/9511026* [quant-ph].
- [204] D. S. Abrams and S. Lloyd, *Phys. Rev. Lett.* **83**, 5162 (1999).
- [205] A. M. Childs, Y. Su, M. C. Tran, N. Wiebe, and S. Zhu, *Phys. Rev. X* **11**, 011020 (2021).
- [206] S. Zhuk, N. Robertson, and S. Bravyi, “Trotter error bounds and dynamic multi-product formulas for hamiltonian simulation,” (2024), *arXiv:2306.12569* [quant-ph].
- [207] S. Keller, K. Boguslawski, T. Janowski, M. Reiher, and P. Pulay, *The Journal of Chemical Physics* **142**, 244104 (2015).
- [208] W. Sennane, J.-P. Piquemal, and M. J. Rančić, *Phys. Rev. A* **107**, 012416 (2023).
- [209] D. Poulin, M. B. Hastings, D. Wecker, N. Wiebe, A. C. Doberty, and M. Troyer, *Quantum Info. Comput.* **15**, 361–384 (2015).
- [210] W. van Dam, M. Mykhailova, and M. Soeken, in *Proceedings of the SC’23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis* (2023) pp. 1414–1419.
- [211] V. von Burg, G. H. Low, T. Häner, D. S. Steiger, M. Reiher, M. Roetteler, and M. Troyer, *Physical Review Research* **3**, 033055 (2021).
- [212] M. Motta, E. Ye, J. R. McClean, Z. Li, A. J. Minnich, R. Babbush, and G. K.-L. Chan, *npj Quantum Information* **7**, 83 (2021).

Appendices

Appendix A: Analysis of logical circuits

for quantum resource estimation

1. Workflows for generating logical circuits

In this section, we outline our workflow for generating the logical quantum circuits that serve as input to the resource estimation pipeline, which computes the associated physical resource requirements. The circuits we generate pertain to quantum simulations for estimating the ground-state energy of molecules. We first specify the quantum simulation algorithm used. We then discuss the workflow for how we obtain the quantum circuits that implement this algorithm from the basic specifications of a molecule. Finally, we analyze the various bounds on the errors incurred in the process of generating the logical circuits, and explain how these bounds need to be chosen to ensure the quantum simulations achieve a given target accuracy. While in our study we use the *p*-benzynes molecule as a concrete example, the described methodology applies to other molecules. Our analysis follows closely the approach of Ref. [35].

The quantum phase estimation (QPE) algorithm [203, 204] is arguably one of the most rigorous quantum computational approaches for estimating ground-state energies in quantum chemistry. Quantum phase estimation is designed to sample in the eigenbasis of the molecular Hamiltonian H by measuring the phase accumulated on an initial input quantum state acted upon by a unitary operator whose eigenvalue spectrum is a function of the spectrum of H . The standard approach is to implement QPE with the time-evolution operator $\exp(-iHt)$. Even more advanced approaches have been proposed; for example, the framework of qubitization [125] allows taking a Hamiltonian given by a sum of unitaries (which is the typical case for quantum chemistry Hamiltonians) and constructing a new operation called “qubitization” that has a functional dependence on the eigenvalues of the Hamiltonian and thus can be used in QPE in place of $\exp(-iHt)$ (see Ref. [133]). Nevertheless, our quantum resource estimation (QRE) analysis pertains to implementing the standard QPE algorithm, that is, we generate concrete QREs for implementing the time-evolution operator $\exp(-iHt)$ by a quantum circuit. More specifically, for a given molecule, we generate QREs for Hamiltonian simulation based on the use of product formulas (PF). In general, an operator $\mathcal{S}_p(t)$ is called an order- p product formula associated with the time-evolution operator $\exp(-iHt)$ for a given Hamiltonian H if (cf. Refs. [205, 206])

$$\mathcal{S}_p(t) = \exp(-iHt) + \mathcal{O}(t^{p+1}). \quad (\text{A1})$$

Our resource estimation analyses are based on using either the first-order Lie–Trotter formula or the second-order Trotter–Suzuki formula, and the resource estimations pertain to implementing a single Trotter slice based on either of these formulas. In the framework of second quantization, the electronic model Hamiltonian is typi-

cally given as

$$\hat{H} = \sum_{p,q} h_{pq} \hat{a}_p^\dagger \hat{a}_q + \frac{1}{2} \sum_{p,q,r,s} h_{pqrs} \hat{a}_p^\dagger \hat{a}_r^\dagger \hat{a}_q \hat{a}_s, \quad (\text{A2})$$

where \hat{a}_p^\dagger and \hat{a}_p are the fermionic creation and annihilation operators, respectively, associated with a given basis set of spin-orbital basis functions $\{\phi_p(\mathbf{x})\}$ (where $\mathbf{x} \equiv \{\mathbf{r}, \sigma\}$ summarizes the orbital and spin degrees of freedom), and the scalar coefficients h_{pq} and h_{pqrs} are the one- and two-electron integrals, respectively, over the basis functions, computed using the kinetic term and the nuclear and electron–electron coulomb potentials. Numerous software tools exist to derive the second-quantized Hamiltonian from the molecular specifications, which include basic information to fully characterize the system, such as the type of participating atoms and the molecule’s geometry (typically summarized in an *xyz* file), total charge, and total spin. For this study, we used Tangelo which is an open source Python software package for end-to-end chemistry workflows for quantum computation [123]. The *p*-benzynes molecule C_6H_4 (which has zero total charge) exhibits a biradical open-shell singlet ground state (it has zero total spin), with two unpaired electrons. Its geometry is specified by the *xyz* configuration shown in table V (cf. Section 19 in the supplementary material of Ref. [207]).

| | | | |
|---|---------|---------|--------|
| C | −0.7396 | −1.1953 | 0.0000 |
| C | 0.7396 | −1.1953 | 0.0000 |
| C | 1.3620 | 0.0000 | 0.0000 |
| C | 0.7396 | 1.1953 | 0.0000 |
| C | −0.7396 | 1.1953 | 0.0000 |
| C | −1.3620 | 0.0000 | 0.0000 |
| H | 1.1999 | −2.1824 | 0.0000 |
| H | −1.1999 | 2.1824 | 0.0000 |
| H | 1.1999 | 2.1824 | 0.0000 |
| H | −1.1999 | −2.1824 | 0.0000 |

Table V: Molecular geometry of *p*-benzynes in ångströms, in terms of the *xyz* file format; cf. Ref. [207].

In addition to the molecule specifications, we need to select a basis set $\{\phi_p(\mathbf{x})\}$. *Basis set selection* can be a challenging task. While theoretically an infinite basis is required to represent the true molecular multi-body wavefunction, in practice we cannot perform calculations using an infinite number of basis functions and must therefore rely on using a finite basis set. Numerous basis sets have been introduced and extensively studied in quantum computational chemistry. The most common minimal basis sets are the ST0-*n*G basis sets, which are derived from a Slater-type orbital basis set, with n denoting the number of Gaussian primitive functions used to represent each Slater-type orbital. While minimal basis sets are computationally inexpensive, they typically result in insufficiently precise computations. Pople basis sets are a type of split-valence basis sets which use more than one basis function to represent valence orbitals, because it is the valence electrons that typically contribute

to the molecular bonding. An entire hierarchy of Pople basis sets have been studied. Importantly, as the basis set grows larger, the resulting approximation gets closer to the true wavefunction; however, increasing the basis set size also results in increasing the required computational resources in space and in time. As a rule of thumb, to achieve semi-quantitative energies, the minimum requirement is to use double-zeta basis sets (such as, e.g., 6-31G or cc-pvdz) [123]. For our QRE analysis, we used the 6-31G basis set; this basis set yields a good trade-off between accuracy and computation time.

Once a basis set has been selected, we can reduce the size of the system (and thus the computational cost) via *active space selection*. This concept relies on the notion that, when only considering a subset of the full active space, the resulting loss in correlations affecting the energy computation can be small. For example, in the so-called “frozen-core approximation”, low-lying occupied core orbitals (which typically do not mix with valence orbitals) are “frozen”, that is, they are not included in the computation. Choosing which molecular orbitals to freeze is not a trivial task. Again, we used Tangelo, which provides a means to identify the active space specified by the numbers of molecular orbitals to be included that are energetically next to (i.e., below or above) the highest occupied molecular orbital (HOMO) and the lowest unoccupied molecular orbital (LUMO). For example, the specification “HOMO−2 and LUMO+1” means that we include two additional molecular orbitals below the HOMO and one additional orbital above the LUMO. A common choice is to employ an equal number of additional orbitals to be included next to the HOMO and LUMO; this choice leads to lower energies as opposed to active spaces with unequal numbers of orbitals next to the HOMO and LUMO (cf. Ref. [208]). For example, for the *p*-benzynes molecule, the full active space involves 68 active spin-orbitals for the ST0-3G basis and 124 active spin-orbitals for the 6-31G, the frozen-core approximation involves 56 active spin-orbitals for the ST0-3G basis and 112 active spin-orbitals for the 6-31G, while, for instance, an active space selection ranging from HOMO−5 to LUMO+5 involves only 24 active spin-orbitals for both basis sets. Note that the number of qubits required to encode the system equals the number of active spin-orbitals.

Once both the basis set and the active space have been selected, we can generate the associated second quantized Hamiltonian, as given in eq. (A2). The last step is to translate the model Hamiltonian from the second quantization framework to a framework suitable for the quantum circuit model. This step uses a *fermion-to-qubit mapping*, which is typically either the Jordan–Wigner or the Bravyi–Kitaev transformation, to obtain the Hamiltonian in the Pauli-product form. For a Hamiltonian

acting on n qubits, it can be expressed as

$$H = \sum_{\ell=1}^L H_{\ell} = \sum_{\ell=1}^L \gamma_{\ell} P^{(\ell)}, \quad \text{where} \\ P^{(\ell)} := P_1^{(\ell)} \otimes P_2^{(\ell)} \dots \otimes P_n^{(\ell)}, \\ P_k^{(\ell)} \in \{I, X, Y, Z\}, \quad (\text{A3})$$

and $\gamma_{\ell} \in \mathbb{R}$ are real coefficients. This Hamiltonian can be directly translated into a quantum circuit implementing a single Trotter slice for a PF associated with the time evolution $\exp(-iHt)$ as part of QPE using well-established quantum circuit decomposition methods. This circuit typically consists of a sequence of single- and two-qubit Clifford gates and L arbitrary-angle single-qubit rotations acting on the qubits involved. This circuit is output as a `qasm` text file and used as input to the QRE pipeline.

2. Analysis of Trotter errors and their propagation into phase estimation

We now discuss the various errors incurred in the process of generating the logical circuits and how we can guarantee quantum simulations within a given target accuracy by satisfying certain error bounds, which in turn must be within given *error budgets*. On the logical level, there are three sources of error in implementing the QPE algorithm with Hamiltonian simulation based on using PFs. The first error source is the actual use of an order- p PF, which results in an additive error $\mathcal{O}(t^{p+1})$ in representing the time evolution $\exp(-iHt)$; the additional Trotterization is a technique for dividing the evolution time into many smaller time steps that reduce this error by a constant referred to as “the number of Trotter steps” (or slices). The second error source is associated with circuit synthesis: each term in a PF is implemented by a circuit consisting of single- and two-qubit Clifford gates (such as H , S , Pauli, and CNOT gates) and some arbitrary-angle, single-qubit rotation $R_Z(\theta_{\ell})$ (with angle θ_{ℓ} related to the coefficient γ_{ℓ} in the Hamiltonian in eq. (A3)). The latter is approximated by some sequence of the form $HTHST^{\dagger} \dots HS$. This approximation is found using circuit synthesis tools based on either the Solovay–Kitaev (SK) algorithm or the Ross–Selinger (RS) algorithm [137, 138] that achieves a more favourable scaling in terms of the T gate count. Either of the circuit synthesis methods incurs an error associated with the approximation. The third error source is associated with the precision of estimating the phase in the actual QPE algorithm. In what follows, we elaborate on these errors and provide useful analytic error bounds, which serve to guarantee that some error budgets in our QRE analysis are satisfied.

Our circuits pertain to the first-order Lie–Trotter formula or the second-order Trotter–Suzuki formula, defined

as [205]

$$\mathcal{S}_1(t) := \prod_{\ell=1}^L \exp(-itH_\ell), \quad (\text{A4})$$

$$\mathcal{S}_2(t) := \left(\prod_{\ell=L}^1 \exp(-itH_\ell/2) \right) \left(\prod_{\ell=1}^L \exp(-itH_\ell/2) \right). \quad (\text{A5})$$

Using Propositions 9 and 10 from Ref. [205], we obtain the following analytic error bounds in terms of the spectral operator norm:

$$\|\mathcal{S}_1(t) - \exp(-itH)\| \leq \frac{t^2}{2} \sum_{l_1=1}^L \left\| \left[\sum_{l_2=l_1+1}^L H_{l_2}, H_{l_1} \right] \right\| \leq t^2 \sum_{l=1}^L \sum_{j=l+1}^L C_{lj} |\gamma_l \gamma_j| \quad (\text{A6})$$

$$\begin{aligned} \|\mathcal{S}_2(t) - \exp(-itH)\| &\leq \frac{t^3}{12} \sum_{l_1=1}^L \left\| \left[\sum_{l_3=l_1+1}^L H_{l_3}, \left[\sum_{l_2=l_1+1}^L H_{l_2}, H_{l_1} \right] \right] \right\| + \frac{t^3}{24} \sum_{l_1=1}^L \left\| \left[H_{l_1}, \left[H_{l_1}, \sum_{l_2=l_1+1}^L H_{l_2} \right] \right] \right\| \\ &\leq \frac{t^3}{3} \sum_{l_1=1}^L \sum_{l_3=l_1+1}^L \sum_{l_2=l_1+1}^L C_{l_1 l_2 l_3} |\gamma_{l_1} \gamma_{l_2} \gamma_{l_3}| + \frac{t^3}{8} \sum_{l_1=1}^L \sum_{l_2=l_1+1}^L C_{l_1 l_2} |\gamma_{l_1}^2 \gamma_{l_2}|, \end{aligned} \quad (\text{A7})$$

where $C_{lj} = 1$ if $[P^{(j)}, P^{(l)}] \neq 0$, and $C_{lj} = 0$ if $[P^{(j)}, P^{(l)}] = 0$; similarly, $C_{l_1 l_2 l_3} = 1$ if $[P^{(l_3)}, [P^{(l_2)}, P^{(l_1)}]] \neq 0$, and $C_{l_1 l_2 l_3} = 0$ otherwise. The first expressions, in terms of commutators, have been proven to be tight bounds for the order-1 and order-2 PFs, respectively [205].

The standard Hamiltonian simulation based on PFs approximates the unitary time evolution by splitting it into r Trotter slices:

$$\exp(-itH) = [\mathcal{S}_p(t/r)]^r + \mathcal{O}(r(t/r)^{p+1}). \quad (\text{A8})$$

The smaller the time $\tau := t/r$ for a single Trotter slice is, the better the approximation associated with Trotterization becomes. Note that $\lim_{r \rightarrow \infty} [\mathcal{S}_p(t/r)]^r = \exp(-itH)$. Each term $U_\ell(\tau) := \exp(-i\tau H_\ell)$ for the order-1 PF (or $U_\ell(\tau) := \exp(-i\tau H_\ell/2)$ for the order-2 PF) is implemented by a circuit consisting of single- and two-qubit Clifford gates along with an additional arbitrary-angle single-qubit rotation; the latter needs to be decomposed and approximated by some sequence of the form $HTHST^\dagger \dots HS$ using the SK algorithm (or the RS algorithm). The incurred error of approximation is required to be bounded by some error budget per gate. More precisely, we let the effective unitary $\tilde{U}_\ell(\tau)$ denote the approximation of $U_\ell(\tau)$ by a circuit consisting of gates from the standard gate set $\{H, S, T, \text{Pauli gates, CNOT}\}$ after running the SK algorithm and using other circuit synthesis tools, and let Δ_{synth} denote the maximum circuit synthesis error in this approximation in terms of the spectral norm. We then

define the *error budget per gate* for the SK algorithm, denoted by δ , to be a given upper bound on the allowable circuit synthesis error:

$$\Delta_{\text{synth}} := \max_{\ell} \|U_\ell(\tau) - \tilde{U}_\ell(\tau)\| \leq \delta. \quad (\text{A9})$$

Similar to the approach in Ref. [35], we define, for any $t \geq 0$, an *effective* Hamiltonian associated with the resulting quantum circuit:

$$\tilde{H}_{\text{eff}}(t) := i \ln \left([\tilde{\mathcal{S}}_p(\tau)]^r \right) / t, \quad \text{where}$$

$$\begin{aligned} \tilde{\mathcal{S}}_1(t) &:= \prod_{\ell=1}^L \tilde{U}_\ell(t), \\ \tilde{\mathcal{S}}_2(t) &:= \left(\prod_{\ell=L}^1 \tilde{U}_\ell(t) \right) \left(\prod_{\ell=1}^L \tilde{U}_\ell(t) \right). \end{aligned} \quad (\text{A10})$$

The operator logarithm is well-defined, because $[\tilde{\mathcal{S}}_p(\tau)]^r$, which is a product of unitary operations, is invertible. The operator $\tilde{H}_{\text{eff}}(t)$ is the effective Hamiltonian associated with the *effective* unitary $\tilde{U}(t) := \exp(-it\tilde{H}_{\text{eff}}(t))$ that symbolically represents the circuit resulting from two procedures: (i) the use of a PF along with Trotterization and (ii) circuit synthesis involving especially the SK algorithm. When we run the QPE algorithm, we use circuits effectively represented by controlled applications of the unitary $\tilde{U}(t)$, that is, the quantum circuit implementation of the QPE algorithm is designed to estimate the energy eigenvalues of the effective

Hamiltonian $\tilde{H}_{\text{eff}}(t)$ (rather than those of H). Since $\tilde{U}(t)$ represents a perfect circuit consisting of gates from the standard gate set, the only additional error incurred is that associated with the accuracy of the actual phase estimation in the inference process of the eigenvalues of $\tilde{H}_{\text{eff}}(t)$.

We aim to bound $|E_0 - E_{\text{eff}(t),0}|$, which is the difference between the ground-state energy E_0 of H (which we aim to estimate) and the lowest eigenvalue of $\tilde{H}_{\text{eff}}(t)$ (which we actually estimate). According to Lemma 3 in the supplementary material of Ref. [35], for any given target error bound ϵ , $\|H - H_{\text{eff}}(t)\| \leq \epsilon$ also implies $|E_0 - E_{\text{eff}(t),0}| \leq \epsilon$. Moreover, according to Lemma 4 in Ref. [35], the assumption that $\|\exp(-itH) - \exp(-itH_{\text{eff}}(t))\| \leq \gamma(t)t$ is true for some nondecreasing continuous function $\gamma(t)$ on $[0, \infty)$ implies $\|H - H_{\text{eff}}(t)\| \leq \gamma(t)$. We can use these implications to deduce a relation between the error in energy estimation and the errors associated with the Trotter–Suzuki approximation and the circuit synthesis as follows. Using the triangle inequality multiple times and the analytic bounds given in Equations (A6) and (A7), we may infer the following:

$$\begin{aligned} \|\exp(-itH) - [\mathcal{S}_p(\tau)]^r\| &= \\ &= \|\exp(-i(t/r)H) - [\mathcal{S}_p(t/r)]^r\| \\ &\leq r \|\exp(-i(t/r)H) - \mathcal{S}_p(t/r)\| \\ &= r \|\exp(-iH\tau) - \mathcal{S}_p(\tau)\| \\ &=: \Delta E_{\text{TS}[p]}(t)t, \end{aligned} \quad (\text{A11})$$

where

$$\Delta E_{\text{TS}[1]}(t) := \tau \sum_{l=1}^L \sum_{j=l+1}^L C_{lj} |\gamma_l \gamma_j|, \quad (\text{A12})$$

$$\begin{aligned} \Delta E_{\text{TS}[2]}(t) &:= \frac{\tau^2}{3} \sum_{l_1=1}^L \sum_{l_3=l_1+1}^L \sum_{l_2=l_1+1}^L C_{l_1 l_2 l_3} |\gamma_{l_1} \gamma_{l_2} \gamma_{l_3}| \\ &+ \frac{\tau^2}{8} \sum_{l_1=1}^L \sum_{l_2=l_1+1}^L C_{l_1 l_2} |\gamma_{l_1}^2 \gamma_{l_2}|. \end{aligned} \quad (\text{A13})$$

Moreover, by repeated use of the triangle inequality, we can prove by induction that

$$\left\| [\mathcal{S}_p(\tau)]^r - [\tilde{\mathcal{S}}_p(\tau)]^r \right\| \leq \begin{cases} rL\Delta_{\text{synth}} & \text{for } p = 1, \\ r(2L-1)\Delta_{\text{synth}} & \text{for } p = 2. \end{cases} \quad (\text{A14})$$

Hence, using the triangle inequality, we may infer the following:

$$\begin{aligned} \left\| \exp(-itH) - [\tilde{\mathcal{S}}_p(\tau)]^r \right\| &\leq \left\| \exp(-itH) - [\mathcal{S}_p(\tau)]^r \right\| + \left\| [\mathcal{S}_p(\tau)]^r - [\tilde{\mathcal{S}}_p(\tau)]^r \right\| \\ &\leq \begin{cases} \Delta E_{\text{TS}[1]}(t)t + rL\Delta_{\text{synth}} & \text{for } p = 1, \\ \Delta E_{\text{TS}[2]}(t)t + r(2L-1)\Delta_{\text{synth}} & \text{for } p = 2. \end{cases} \end{aligned} \quad (\text{A15})$$

Thus, according to Lemmas 3 and 4 and Theorem 1 in the supplementary material of Ref. [35], we can conclude

that the error in the ground-state energy that results from such a simulation is at most

$$|E_0 - E_{\text{eff}(t),0}| \leq \begin{cases} \Delta E_{\text{TS}[1]}(t) + rL\Delta_{\text{synth}}/t & \text{for order-1 PF,} \\ \Delta E_{\text{TS}[2]}(t) + r(2L-1)\Delta_{\text{synth}}/t & \text{for order-2 PF.} \end{cases} \quad (\text{A16})$$

Similar to the approach used in Ref. [35], we define $\epsilon_1 := \Delta E_{\text{TS}[p]}$, $\epsilon_2 := rL\Delta_{\text{synth}}/t$ or $\epsilon_2 := r(2L-1)\Delta_{\text{synth}}/t$ depending on whether we use the first-order or second-order PF, and ϵ_3 to be the error in phase estimation. For chemical significance, the total overall target error $\epsilon := \epsilon_1 + \epsilon_2 + \epsilon_3$ should be at most 0.1 millihartrees, that is, our ideal overall error budget is $\epsilon = 10^{-4}$ hartrees.

The split of the total error budget into three parts is non-trivial; in our QRE analysis, we have treated ϵ_1 , ϵ_2 , and ϵ_3 as parameters and optimized the error budget allocation to these three parts so as to minimize the expected T gate count.

Finally, to determine an appropriate evolution time for the unitary $\exp(-itH)$, we require that the phase

that we estimate using QPE, $\theta := E_0 t$ (where E_0 is the ground-state energy), is within $[0, 2\pi]$. Since we do not have knowledge of the eigenvalues of H , we require that $\|H\|t \leq 2\pi$. As we do not have knowledge of the spectral norm of the Hamiltonian (which is equal to the largest eigenvalue) either, we use $\Gamma := \sum_l |\gamma_l| \geq \|H\|$ and choose $t = 2\pi/\Gamma \leq 2\pi/\|H\|$. This choice of t implies that, when using the first-order Lie–Trotter formula, the number of Trotter slices is given by

$$r = \max \left\{ 1, \left\lceil \frac{\pi \sum_{\ell=1}^L \sum_{j=\ell+1}^L C_{\ell j} |\gamma_\ell \gamma_j|}{\epsilon_1 \sum_{\ell=1}^L |\gamma_{\ell=1}|} \right\rceil \right\}, \quad (\text{A17})$$

which directly follows from Equation (A12). A similar expression for r can be derived when using the second-order Trotter–Suzuki formula by using Equation (A13). The parameters r and L determine the size of the logical quantum circuits. From the knowledge of r and L , we can also infer the error budget per gate for the SK algorithm as defined in Equation (A9), that is,

$$\Delta_{\text{synth}} \leq \delta := \begin{cases} 2\pi\epsilon_2/(rL\Gamma) & \text{for order-1 PF,} \\ 2\pi\epsilon_2/[r(2L-1)\Gamma] & \text{for order-2 PF.} \end{cases} \quad (\text{A18})$$

Estimations of Trotter errors via rigorous analytic upper bounds can be loose, which can result in overestimating the number of Trotter slices by many orders of magnitude. For this reason, several recent studies instead have attempted to predict the number of Trotter slices practically required using various heuristics, such as those based on Monte Carlo sampling. Following this trend, we have conducted an additional empirical QRE analysis based on more-realistic Trotter numbers that we inferred through extrapolation. More concretely, we empirically computed the Trotter error $\|\exp(-itH) - [\mathcal{S}_2(\tau)]^r\|$ via full numerical computations for p -benzyne Hamiltonians pertaining to small active spaces $\text{HL}\pm n$ for $n = 0, 1, 2$. We inferred the corresponding required evolution time τ for a single Trotter slice and the associated Trotter number $\beta := 1/\tau$ to satisfy an error budget associated with a constant accuracy in the energy estimation. Based on the obtained data, we then inferred the approximate scaling of β as a function of the number of active spin orbitals. We found the inferred scaling to be consistent with the results of a prior empirical study based on Monte Carlo sampling [209]. Based on the deduced scaling, we have estimated β values for larger active spaces via regression.

3. Propagation of errors in qubitization

In the qubitization approach, there are three main sources of error, as in [127]. The first arises from truncating the small eigenvalues of the double-factorized Hamiltonian, but by keeping track of the value of the truncated eigenvalues it is possible to bound the 2-norm difference between the original Hamiltonian and the truncated Hamiltonian. The second source of error arises

from approximations in the implementation of the qubitization operator, and can be decomposed into two terms. Namely, when approximating coefficients used during the summing of operators in the LCU decompositions, as well as in the rotation angles in the diagonalization operations in the innermost decomposition of the double factorized Hamiltonian with finite bits of precision. However, by increasing the number of bits used to implement these operations the error terms can be driven to zero. The final contribution to the error is the imprecision in the QPE, as in the PF approach, but this error and requisite number of logical ancillary qubits can be bounded using the total number of repetitions of the qubitization operator.

At the logical level, it is possible to breakdown the error in QPE of the qubitization approach as in Equation (23) of [64]. Namely, the output energy of the QPE is within

$$\Delta E \leq \lambda \sqrt{\left(\frac{\pi}{2^m}\right)^2 + (\epsilon_H + \pi\epsilon_{\text{QPE}})^2} \quad (\text{A19})$$

of the Hamiltonian used in the qubitization approach, where λ is the 1-norm of the Hamiltonian, m is the number of bits in the QPE, ϵ_{QPE} is related to the error in implementing the QPE and is usually negligible, and ϵ_H is related to the 2-norm difference between the qubitization operators of the double factorized Hamiltonian and its implementation.

Following the analysis from [64], we can bound the error contributed from the finite bit precision approximations. The error in the qubitization operator is given by

$$\begin{aligned} \epsilon &\leq \|e^{i \arccos(H/\lambda)} - e^{i \arccos(\tilde{H}/\lambda)}\| \\ &\leq \|\arccos(H/\lambda) - \arccos(\tilde{H}/\lambda)\| \\ &\leq \sum_{p=0}^{\infty} \frac{(2p-1)!!}{\lambda^{2p+1}(2p+1)(2p)!!} \|H^{2p+1} - \tilde{H}^{2p+1}\| \quad (\text{A20}) \end{aligned}$$

The term $\|H^{2p+1} - \tilde{H}^{2p+1}\|$ can be bounded by $(2p+1)(\|H - \tilde{H}\| + \|H\|)^{2p}(\|H - \tilde{H}\|)$, giving us

$$\begin{aligned} \epsilon &\leq \sum_{p=0}^{\infty} \frac{(2p-1)!!}{\lambda^{2p+1}(2p+1)(2p)!!} (2p+1)(\|H - \tilde{H}\|)^{2p+1} \\ &\leq \frac{\|H - \tilde{H}\|}{\lambda} \left[1 - \left(\frac{\|H\| + \|H - \tilde{H}\|}{\lambda} \right)^2 \right]^{1/2}. \quad (\text{A21}) \end{aligned}$$

Let $\Gamma \equiv \|H - \tilde{H}\|$. Solving for Γ in the above inequality, we obtain

$$\Gamma \leq \frac{\sqrt{2\Delta E}}{4\left(1 + \frac{\Delta E^2}{8\lambda^2}\right)} \left(1 - \frac{\|H\|^2}{\lambda^2}\right), \quad (\text{A22})$$

Thus, if we aim the error contribution from implementing qubitization to be within ΔE , then we must choose the bit precisions to be large enough that the 2-norm difference satisfies the above.

Appendix B: Quantum resource estimations using TopQAD

Once a target logical quantum circuit has been generated, we construct a fault-tolerant architecture that can implement this circuit to conduct a QRE analysis using the TopQAD toolkit [99]. The software’s approach to creating a fault-tolerant architecture is described in Ref. [50]. For a given quantum circuit and success probability, we generate an architecture that would feasibly run the computation at the requisite precision. This architecture is one that allows us to estimate the resources required for a specific quantum circuit using hardware that can implement a rotated surface code layout. By abstracting the hardware away, we are able to focus on the layout and from there construct an architecture that can be used to implement those operations required for FTQC.

1. The compilation process

The main idea behind this construction is to transform the given circuit into an optimized sequence of $\pi/8$ Pauli rotations, and then to process these rotations using multi-qubit lattice surgery to connect distant qubits [28, 49, 51]. These $\pi/8$ Pauli rotations can then be implemented on the underlying architecture, where magic states are distilled and consumed through specific applications of lattice surgery. This procedure results in a very nontrivial simultaneity condition, as the bus qubits used in the lattice surgeries can only be used for a single rotation at a given point in time. Additionally, there are several conditions for how the bus qubits can interact with qubits storing data for the circuit, leading to complications in the underlying architecture. However, once these conditions have been taken into account, various classical scheduling processes can be used to generate the necessary schedule of operations that we then implement via lattice surgery.

The pipeline followed to generate the QREs for a given quantum circuit and a target success probability starts by transforming the circuit into one in which only Clifford and T gates are used. This requires some algorithm to decompose an arbitrary gate into known elements. The most well-known of such procedures is an implementation of the SK theorem, which, while efficient in a complexity theoretic sense, is actually quite costly in practice. A different procedure with slightly less applicability is the RS algorithm, which results in significantly shorter circuits. Additionally, such implementations quickly become a bottleneck in terms of the reachable error rates, as

the per-gate error budgets for the SK algorithm quickly approach machine precision.

Given that the architecture considered requires that quantum operations are represented by Pauli rotations, the next step is to *transpile* the circuit consisting of Clifford and T gates into a circuit consisting of Pauli rotations. The circuit described in the Clifford + T gate set is first converted to a sequence of $\pi/4$ (Clifford) and $\pi/8$ (non-Clifford) Pauli rotations according to the conversion rules described in Ref. [28]. After conversion, a procedure is run to remove the Clifford operations from the circuit using commutation rules, leaving only $\pi/8$ rotations. This procedure can be run efficiently using the symplectic representation of Clifford gates [51]. Additionally, since commutable $\pi/8$ rotations can be reordered such that adjacent $\pi/8$ rotations with same axis of rotation can be combined, this allows us to perform operations corresponding to multiple rotation commutations in a single step effectively reducing the T count. As discussed in Ref. [51], this transpilation procedure drastically decreases the overall running time of the circuit even if the resulting circuit makes the operations less parallelizable due to its increased density.

2. The assembly process

At this point, we have constructed a logical circuit tailored to an implementation on a surface code encoding logical qubits. The next step in the pipeline to generate the QREs is the assemble of the structures required for FTQC that will allow the scheduling of the $\pi/8$ Pauli rotations in the circuit. As illustrated in fig. 14, the architecture considered for the scheduling of the logical operations features a core processor, comprising a memory fabric with two-tile two-qubit patches of data qubits and an auto-correcting buffer, which is connected to the MSF using bus qubits, mirroring the configuration used in Ref. [28].

Central to our approach is the utilization of a multi-level MSF for magic state distillation, where the fidelity of magic states undergoes iterative enhancement across successive distillation levels. Low-fidelity magic states are created from operations on physical qubits at magic state preparation units following a magic state preparation protocol [97, 98] as described in III E. Then, at each distillation level, the MSF used the lower-fidelity magic states to create higher-fidelity magic states that are dispatched to a dedicated area where magic states can be enlarged to the required code distance that interfaces with the next round of distillation. A 15:1 distillation protocol is assumed to be used by the distillation units at all levels due to its capacity to improve magic state fidelity in $O(P_T^3)$, where P_T is the logical error rate for input magic states [49]. The uppermost level of the MSF connects to the memory fabric via a buffer space that allows magic states to be temporarily stored before being consumed within the memory fabric. This

space is designed to incorporate auto-correcting buffers, named for their capability to execute corrective measures concurrently with magic state consumption, notably enabling the auto-correcting of $\pi/8$ operations. The auto-correcting buffers and the memory fabric comprise the core processor of the device.

The scheduling methodology employed in the studied topological architecture is presented in Ref. [51] and addresses the sequencing of operations and the allocation of logical resources required for establishing connections between distant qubits in the core processor as needed. Due to the reduced parallelization potential of the $\pi/8$ operations, we assume a serial scheduling is employed in the QREs presented here. If nonrestrictive availability of magic states is ensured, and considering that the expected time to execute a $\pi/8$ rotation is equal to one logical cycle because of the auto-correcting buffers, the minimum number of logical cycles required to execute the entire circuit in a serial scheduling is equal to its T count, ignoring the warm-up time. Each logical cycle requires performing d_{core} parity checks, where d_{core} is the code distance of the logical qubits in the core processor, each taking a time $T_M + 4T_2 + 2T_1 + T_M + t_R$, considering the measurement time t_M , the reset time t_R , the single-qubit gate time t_1 , and the two-qubit gate time t_2 . Therefore, while it is easy to generate time estimates for circuits scheduled in serial, generating the space estimates require creating a MSF with enough distillation units capable of distilling magic states quickly enough to keep the core processor constantly busy.

The assembler of the quantum architecture described requires minimizing the space (i.e., the physical qubits required) under a given error budget. The decisions to be made are related to sizing the components of the architecture, i.e., the core processor and the MSF, while ensuring fault tolerance. The given error budget is distributed between errors that arise in the execution of quantum operations in the core processor, E_{core} , and in the distillation of magic states in the MSF, E_{MSF} . Therefore,

$$E_{\text{core}} + E_{\text{msf}} \leq E. \quad (\text{B1})$$

The errors of the core processor and the MSF are modeled and predicted following the pipeline described in [50]. Since we provide QREs for the case with a never idling core processor, the accumulated errors in the core are only resulting from the Clifford operations required for the multi-qubit lattice surgeries performed and the protection of the idling data qubits while lattice surgeries involving other data qubits are occurring in the core. The accumulated errors in the core processor is approximated as

$$E_{\text{core}} \approx (2Q + \sqrt{8Q} + 29)Te_{\text{mem,core}}. \quad (\text{B2})$$

which assumes that all $(2Q + \sqrt{8Q} + 29)$ logical qubits in the core processor (approximated size of the memory fabric and buffer) following the design presented in

Figure 14, where Q is the number of data qubits in the circuit, are susceptible to result in an error with probability $e_{\text{mem,core}}$ during all the T logical cycles required to run the circuit. The error rate $e_{\text{mem,core}}$ is derived from emulations of the FTQC protocol for quantum memory. These emulations establish the correlation between code distance and logical error rates based on a given choice of physical parameters, resulting in a predictive model obtained by regression from numerical simulations at low code distances, using efficient stabilizer circuit simulators [102] following the descriptions in Section III B. In the MSF, the error rate of output magic states is resulting from the preparation, distillation and expansion procedures. Following [50], considering e_{prep} as the error rate for the magic states prepared from physical qubits and that the Clifford and growth accumulated errors can be approximated to the memory errors, i.e., $e_{\text{cliff}} = e_{\text{grow}} = e_{\text{mem}}$, the magic state error rates of the entire MSF using 15:1 distillation units can be calculated recursively as follows:

- Input to level 1: $e_{\text{in},1} = e_{\text{prep}}$;
- Output from level l for all $l \in \{1, \dots, L\}$:

$$e_{\text{out},l} = 35e_{\text{in},l}^3 + 7.1e_{\text{mem},l};$$

- Input to level $l+1$ for all $l \in \{1, \dots, L\}$:

$$e_{\text{in},l+1} = 1 - (1 - e_{\text{out},l})(1 - e_{\text{mem},l}).$$

Therefore, the error rate of the magic state input to the core processor is $e_{\text{core}} = e_{\text{in},L+1}$ for an MSF with L distillation levels. Given that T magic states needs to be distilled for the entire execution of the quantum program, we have that

$$E_{\text{msf}} = e_{\text{core}}T. \quad (\text{B3})$$

The choice of hardware parameters to determine the required number of distillation levels L and code distances $d_l, \forall l \in 1, \dots, L+1$, which includes the core processor as $l = L+1$, is such that it must reach the target logical error rates based on Equations (B1) to (B3). The process followed to make these decisions is described in Ref. [50]. In summary, the core processor's code distance d_{L+1} is minimized, assuming $E_{\text{msf}} = 0$. Then, it sets the first level code distance d_1 considering the residual error budget left after the core level logical encoding is decided, i.e., $E_{\text{msf}} \leq E - E_{\text{core}}$. Next, it determines the number of distillation levels L required to meet the magic state error rate requirement e_{core} derived from B3 for the residual error budget. Finally, if $L > 1$, it calculates the code distances d_l for all levels that can meet the error budget using the minimum number of physical qubits across the whole architecture. Once these decisions have been made, the assembler determines the number of distillation units required for a steady flow of magic states to the core processor such that the distillation rate of magic

states output from the MSF matches the consumption rate of magic states in the core processor.

While it is possible for each distillation level to contain only a single distillation unit, such a configuration introduces significant idling time, thereby prolonging the expected runtime of executing quantum circuits in our proposed architecture. In addition, although having fewer units implies that there are fewer logical qubits, this solution potentially increases physical space requirements due to the larger code distances resulting from the additional overhead incurred from logical operations executed on data qubits to mitigate decoherence during idling time [50].

3. Comparison with the AzureQRE toolkit

To provide additional logical and physical resource estimates, we use the Azure Quantum Resource Estimator (AzureQRE) [112, 210]. We used the surface code option within AzureQRE and the hardware parameters of Table I and estimate the resources required only for the double-factorized qubitization algorithm. We refer the reader to Refs. [210] and [49] for full details about the architectural assumptions, but briefly highlight AzureQRE assumes a 2D nearest-neighbor layout which has the ability to perform parallel operations and utilizes 15-to-1 magic state distillation. We show the results for physical resource estimates in Table VI, where we have also included the results from TopQAD’s resource estimates for the DF qubitization algorithm in Table IV for easy comparison. Compared with the estimates based on TopQAD presented in the main text (see Section IV), AzureQRE finds that the baseline parameter set is not below the surface code threshold, specifically because of the measurement error rate. The TopQAD suite’s QRE pipeline includes more-advanced FTQC protocols, including magic state factories and space-time trade-offs [50] compared with those implemented in AzureQRE. For the target and desired hardware parameter sets, TopQAD’s estimates are about an order of magnitude lower in terms of the number of physical qubits and by around a factor of 3 lower in terms of the runtime than AzureQRE, due to its use of more advanced FTQC protocols.

Appendix C: Double-factorized quantum chemistry

The standard quantum chemistry Hamiltonian is

$$H = \sum_{ij,\sigma} h_{ij} a_{i\sigma}^\dagger a_{j\sigma} + \frac{1}{2} \sum_{ijkl,\sigma\rho} h_{ijkl} a_{i\sigma}^\dagger a_{j\rho}^\dagger a_{k\rho} a_{l\sigma}, \quad (\text{C1})$$

where h_{ij} and h_{ijkl} are the one- and two-electron integrals, σ and ρ index spin, and $a_{p\sigma}$ are fermionic creation and annihilation operators. To implement the time evolution of this Hamiltonian on a quantum computer, double-factorization can be used as a resource-efficient alternative to Trotterization [211].

The fourth-order Coulomb tensor h_{ijkl} can be written as a $N_o^2/4 \times N_o^2/4$ electronic repulsion integral (ERI) matrix, A , where N_o is the number of *spin orbitals*. A is positive semi-definite and generally has a rank $L = \mathcal{O}(N)$ for chemical systems. We can diagonalize A , leading to a decomposition in terms of an auxiliary tensor \mathcal{L} such that [212]:

$$A = \sum_{\ell=0}^{L-1} (\mathcal{L}^{(\ell)})^2 = \sum_{\ell=0}^{L-1} \sum_{ijkl=0}^{N/2-1} \mathcal{L}_{ik}^{(\ell)} \mathcal{L}_{jl}^{(\ell)} a_i^\dagger a_k a_j^\dagger a_l. \quad (\text{C2})$$

Each matrix $\mathcal{L}^{(\ell)}$ can then be further decomposed, giving a set of eigenvalues $\{\lambda_m^{(\ell)}\}$ and a diagonalizing unitary $U^{(\ell)}$. This leads to the double-factorized form of the Hamiltonian H_{DF} :

$$H_{\text{DF}} = \sum_{ij,\sigma} \tilde{h}_{ij} a_{i\sigma}^\dagger a_{j\sigma} + \frac{1}{2} \sum_{\ell=0}^{L-1} \left(\sum_{ij,\sigma} \sum_m \lambda_m^{(\ell)} U_{m,i}^{(\ell)} U_{m,j}^{(\ell)} a_{i\sigma}^\dagger a_{j\sigma} \right)^2, \quad (\text{C3})$$

where $\tilde{h}_{ij} \equiv h_{ij} - \frac{1}{2} \sum_l h_{iill}$ comes from the reordering of the creation and annihilation operators. By truncating some of the eigenvalues, a low-rank approximation can be obtained. There exists efficient walk operators, W , which implement this Hamiltonian as a quantum circuit, as described in Ref. [211].

Appendix D: Runtime of classical algorithms for quantum chemistry

To provide realistic estimates of the classical resources required for various classical quantum chemistry algorithms, we extrapolate the results of recent publications that use full configuration interaction (FCI) [140] and density matrix renormalization group (DMRG) [130]. For the FCI calculations, we note that Ref. [140] reports running their largest system of C_3H_8 in an STO-3G basis, which has 26 electrons in 23 orbitals, a calculation involving 1.3 trillion determinants, took 113.6 hours using 512 processes, a total of around 58k CPU hours. Assuming quadratic scaling with number of determinants ($\mathcal{O}(N_{\text{det}}^2)$) scaling for the FCI algorithm, we use this single data point to compute a realistic prefactor for the computational time scaling. Note that the number of determinants scales exponentially with number of orbitals. We then take the worst-case number of determinants for each number of orbitals, where the number of electrons (N_e) is equal to the number of orbitals (N_o), and calculate the total number of determinants as $N_{\text{det}} = (N_o! / (N_o - N_e)! N_e!)^2$ and, assuming a factor of 1000 in parallelism, compute the time for various numbers of orbitals, consistent with the 512 CPUs used in Ref. [140].

| | | Baseline Parameter Set ($\Lambda \approx 2.34$) | | | | Target Parameter Set ($\Lambda \approx 9.3$) | | | Desired Parameter Set ($\Lambda \approx 18$) | | |
|-------------------------|------------------|---|-------------------|--------------------|------------------|--|--------------------|----------------|--|--------------------|---------|
| Target error ϵ | N_{orb} | # Phys. qubits | Phys. time | QEC code distances | # Phys. qubits | Phys. time | QEC code distances | # Phys. qubits | Phys. time | QEC code distances | |
| AzureQRE | 1.0 mHa | 6 | - | - | - | 1.2×10^7 | 28.8 minutes | 77 | 4.1×10^6 | 16.1 minutes | 43 |
| | | 18 | - | - | - | 2.4×10^7 | 1.2 days | 91 | 7.8×10^6 | 15.7 hours | 51 |
| | | 26 | - | - | - | 3.3×10^7 | 4.8 days | 95 | 1.0×10^7 | 2.7 days | 53 |
| | | 76 | - | - | - | 1.0×10^8 | 1.4 years | 111 | 3.2×10^7 | 289.0 days | 63 |
| | 0.1 mHa | 6 | - | - | - | 1.6×10^7 | 6.1 hours | 85 | 5.0×10^6 | 3.4 hours | 47 |
| | | 18 | - | - | - | 3.0×10^7 | 15.0 days | 97 | 9.4×10^6 | 8.5 days | 55 |
| | | 26 | - | - | - | 4.1×10^7 | 62.1 days | 103 | 1.3×10^7 | 34.4 days | 57 |
| | | 76 | - | - | - | 1.3×10^8 | 21.0 years | 119 | 4.0×10^7 | 11.8 years | 67 |
| TopQAD | 1.0 mHa | 6 | 1.5×10^7 | 5.4 minutes | 25, 61 75 | 1.6×10^6 | 1.2 minutes | 23 27 | 8.9×10^5 | 57.6 seconds | 17 21 |
| | | 18 | 4.9×10^7 | 2.2 days | 15, 33, 75 91 | 4.0×10^6 | 12.4 hours | 11, 29 33 | 2.1×10^6 | 10.2 hours | 21 27 |
| | | 26 | 6.0×10^7 | 10.0 days | 15, 33, 87 103 | 5.5×10^6 | 2.3 days | 11, 31 37 | 2.8×10^6 | 1.7 days | 23 27 |
| | | 76 | 1.2×10^8 | 2.6 years | 17, 37, 93 109 | 1.5×10^7 | 234.1 days | 13, 33 43 | 8.0×10^6 | 168.8 days | 27 31 |
| | 0.1 mHa | 6 | 2.3×10^7 | 12.6 hours | 29, 77 91 | 3.0×10^6 | 2.7 hours | 11, 27 31 | 1.4×10^6 | 2.2 hours | 21 25 |
| | | 18 | 5.8×10^7 | 30.8 days | 15, 35, 91 105 | 5.1×10^6 | 6.5 days | 13, 31 35 | 2.6×10^6 | 5.4 days | 23 29 |
| | | 26 | 7.9×10^7 | 132.1 days | 19, 35, 85 115 | 6.8×10^6 | 28.5 days | 13, 31 39 | 3.4×10^6 | 21.2 days | 25 29 |
| | | 76 | 1.5×10^8 | 28.5 years | 17, 41, 99 119 | 1.8×10^7 | 6.5 years | 15, 35 43 | 9.8×10^6 | 5.0 years | 29 33 |

Table VI: Physical resource estimates generated by TopQAD [99] and AzureQRE [112] for implementing the QPE algorithm on electronic-structure quantum circuits associated with the *p*-benzine and FeMoco molecules, for two precisions in energy estimation: qualitatively accurate computation within a target error 1.0 mHa, and quantitatively accurate computation within a target error 0.1 mHa, respectively, using a circuit-level error budget of 0.01 using the double-factorized qubitization algorithm. We report estimates for the physical wall-clock time and the number of physical qubits required for fault-tolerant implementations of the QPE algorithm for electronic spectra associated with various molecular active spaces with sizes specified by the number of orbitals N_{orb} . The data for $N_{\text{orb}} = 6, 18, 26$ correspond to active space selections HL±2, 8, 12 (using HL±*n* to denote “HOMO−*n* and LUMO+*n*”; see Appendix A 1 for an explanation of these terms) for *p*-benzine using the 6-31G basis to represent the fermionic orbitals; the data for $N_{\text{orb}} = 76$ pertains to the active-space model for FeMoco proposed in Ref. [36]. In addition, we also report the QEC code distances that are required for running the corresponding circuits fault-tolerantly. Here, physical resources are reported only for the quantum circuits based on running the DF qubitization algorithm. The associated resource requirements are reported for three hardware specifications, namely, baseline, target, and desired hardware (Λ_{18} model), as summarized in Table I. Note that the symbol “-” represents that AzureQRE estimates that the baseline parameter set is above the QEC threshold.

For the DMRG calculations, we assume cubic scaling with bond dimension ($\mathcal{O}(\chi^3)$). Note that there is no definitive scaling of bond dimension χ with number of orbitals for generic quantum chemistry problems, but it is generally expected to scale exponentially for strongly correlated systems. Ref. [130] reports estimates of the necessary bond dimension for various homogeneous catalysts as well as runtimes for smaller bond dimension DMRG calculations. Using the data in Table 3 of Ref. [130], specifically the data which was run on a computer cluster, we fit parameters a and b in the scaling function $f(\chi) = a\chi^3 + b$ and then use those coefficients to predict the runtime necessary for the reported bond dimensions necessary to reach chemical accuracy, assuming a factor of 100 parallelism, consistent with the 40 CPUs used in the Ref [130].

Appendix E: Circuit-level noise model

We employ a circuit-level depolarizing noise model for the benchmarking and sensitivity analysis simulations in Section III B. It consists of three types of errors: gate errors, idling errors, and state preparation and measurement (SPAM) errors, where the strength of each type is

determined from the hardware noise parameters, such as those found in Table I.

Imperfect gates are modeled by adding a depolarizing noise channel at rate p to the gate qubits after the application of each gate. For one-qubit gates, the noise channel randomly applies one of X , Y or Z , each with probability $p/3$. Similarly, for two-qubit gates, the noise channel applies one of the 15 two-qubit non-identity Pauli gates, each with probability $p/15$. The rate p is determined by utilizing the formula for the depolarizing channel’s average gate fidelity,

$$F_{\text{dep},n} = 1 - \frac{(2^n - 1)2^n}{2^{2n} - 1}p, \quad (\text{E1})$$

where n is the number of gate qubits.

Any qubit that is idling experiences an error, dependent on both the decoherence time T_1 of the qubit and the time t it takes to apply the gate(s) to the active qubits. The error is modeled with a single-qubit depolarizing noise channel with rate equal to

$$p = \frac{3}{4} \left[1 - \exp\left(-\frac{t}{T_1}\right) \right]. \quad (\text{E2})$$

The errors in state preparation and reset are captured by assuming that with rate p the orthogonal state is produced, i.e., $|0\rangle$ is prepared instead of $|1\rangle$ and vice versa.

Similarly, measurements in the Z basis are flipped at rate p . In all three cases the fidelity of the operation is from which we directly determine the rate $p = 1 - F_{\text{SPAM}}$.

$$F_{\text{SPAM}} = \frac{P(0|0) + P(1|1)}{2}, \quad (\text{E3})$$