# How to Build a Quantum Supercomputer:
# Scaling from Hundreds to Millions of Qubits

Masoud Mohseni,[1, *] Artur Scherer,[2] K. Grace Johnson,[1] Oded Wertheim,[3] Matthew Otten,[4] Navid Anjum Aadit,[5] Yuri Alexeev,[6] Kirk M. Bresniker,[7] Kerem Y. Camsari,[5] Barbara Chapman,[8] Soumitra Chatterjee,[8] Gebremedhin A. Dagnew,[2] Aniello Esposito,[7] Farah Fahim,[9] Marco Fiorentino,[7] Archit Gajjar,[1] Abdullah Khalid,[2] Xiangzhou Kong,[2] Bohdan Kulchytskyy,[2] Elica Kyoseva,[6] Ruoyu Li,[10] P. Aaron Lott,[11, 12] Igor L. Markov,[13] Robert F. McDermott,[4, 14] Giacomo Pedretti,[1] Pooja Rao,[6] Eleanor Rieffel,[12] Allyson Silva,[2] John Sorebo,[13] Panagiotis Spentzouris,[9] Ziv Steiner,[3] Boyan Torosov,[2] Davide Venturelli,[11, 12] Robert J. Visser,[10] Zak Webb,[2] Xin Zhan,[1] Yonatan Cohen,[3] Pooya Ronagh,[2, 15, 16, 17] Alan Ho,[14] Raymond G. Beausoleil,[1] and John M. Martinis[14, †]

[1]*Emergent Machine Intelligence, 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]*Quantum Algorithm Engineering, Nvidia Corporation, Santa Clara, CA, USA*
[7]*Hewlett Packard Labs, CA, USA*
[8]*Hewlett Packard Enterprise, TX, USA*
[9]*Fermi National Accelerator Laboratory, IL, USA*
[10]*Applied Materials, CA, USA*
[11]*USRA Research Institute for Advanced Computer Science, CA, USA*
[12]*Quantum Artificial Intelligence Laboratory (QuAIL),*
*NASA Ames Research Center, CA, USA*
[13]*Synopsys, CA, USA*
[14]*Qolab, WI, USA*
[15]*Institute for Quantum Computing, University of Waterloo, ON, Canada*
[16]*Department of Physics & Astronomy, University of Waterloo, ON, Canada*
[17]*Perimeter Institute for Theoretical Physics, ON, Canada*
(Dated: February 3, 2025)

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 systems 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 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 that 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, heterogeneous quantum-probabilistic computing with custom-designed accelerators should be considered as a complementary path toward scalability.

**CONTENTS**

* masoud.mohseni@hpe.com
† john@qolab.ai

## I. INTRODUCTION: PAST, PRESENT, AND FUTURE CHALLENGES OF BUILDING QUANTUM COMPUTERS

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 *decoherence*, 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].

Recently, the high-performance computing (HPC) community has shown significant interest 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 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.

Practical quantum–HPC integration runs into a number of challenges. 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 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 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 performing 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 substantial overhead in physical resources, whose estimation and validation is a critical step toward practical, cost-efficient realization of FTQC at utility scale.

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 from single- and few-qubit systems. Recent advances enabled large enough quantum computers to test system performance and demonstrate the execution of simple quantum algorithms. Experiments have demonstrated logical error rates at the $10^{-10}$ level, although surpassing this rate appears difficult due to errors induced via cosmic rays [16], suggesting further investigation. Despite the large footprint of superconducting qubits, in this position paper we describe how modern semiconductor processing facilitates their scaling.

While we introduce a definitive 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 that these ideas will greatly enhance the likelihood of making useful quantum computers.

### A. Systems engineering for quantum hardware

Systems engineering simultaneously optimizes many system parameters. In contrast, prior research often treats quantum computation as a custom-configured quantum physics experiment focusing on isolated physical phenomena to fully understand each of them. For example, reported metrics often emphasize the best-case quality of a given approach, whereas systems engineering is typically sensitive to average and worst-case quality.

Each technology is sensitive to a different set of systems engineering parameters. 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 the number of qubits 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 $\sim$100 µs, often limited by the time scale of mechanical motions of atoms or ions. In addition, superconducting gates are operated in parallel, whereas ion-trap systems have their primitive gates often processed serially through interaction zones in leading quantum-charge-coupled-device (QCCD) architectures. For NISQ applications with short circuits, repeated trials are typically necessary to obtain sufficient statistics, and thus end-to-end experiments are slow even for superconducting qubits. Generally, one can understand the need for speed by noting that a 1000$\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. Neutral-atom and trapped-ion systems often advertise all-to-all connectivity, but this might not scale beyond small systems, e.g., single traps. 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 less likely to suffer from 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 $\sim$100 for NISQ to $\sim$$10^7$ for utility-scale FTQC (corresponding to $\sim$$10^4$ logical qubits, given QEC overheads), challenges of 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, a comprehensive multi-scale roadmap for superconducting qubits is needed that tackles these challenges and outlines appropriate 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 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, the execution of 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 (30–100$\times$) 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 $\sim 60\,\mu s$ to decode $d = 7$ surface codes [26]. Smaller fast-feedback experiments have demonstrated a decoding response times of $9.6\,\mu s$ [27]. However, at $\sim 0.5$-µs-long stabilization rounds, the total latency inclusive of decoding and redirecting the waveform in an FPGA needs to be within $\sim 5$–$20\,\mu s$ for code distances obtained in our resource estimation studies (see Table V) 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 C 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 address 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, with qubit counts below 1000 in the near future , 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, 10–100$\times$ more qubits can fit into a single dilution refrigerator. This will allow the packing of 20k qubits on a single 14$\times$14-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 B). 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.

**Cryogenic thermal and space budgets.** A key systems engineering challenge is to design the components in a way that it fits within the thermal and space budgets afforded by the cryogenic system. In general, the lower the temperature stage within the cryostat is, the less cooling power and space there is available for the various components. The proposed approach carefully systems engineers the various components accounting for this constraint, balancing qubit quality, complexity of the components, and complexity of fabrication. These constraints become especially acute when scaling from 1000 to 10k qubits. Detailed power and space budgets are not provided in this paper due to proprietary designs.

**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 discover-ing 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 [37]. 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, that is, those with high encoding rates, such as the quantum LDPC codes [38].

A promising approach to reducing the FTQC overhead at this scale is adopting partially fault-tolerant compilation of quantum algorithms, for example, the compilation scheme based on error-corrected Clifford gates and space–time-efficient analog rotations [39, 40]. This approach entirely omits the costly magic state distillation procedures that are typically required for a reliable implementation of non-Clifford gates; it also avoids decompositions of arbitrary-angle rotation gates into the Clifford+$T$ gate set via the Solovay–Kitaev algorithm or alternative techniques (see Section IV and Appendix A 2), which typically result in a large number of $T$ gates. Instead, arbitrary-angle rotation gates are directly executed through analog rotation and ancilla state injection followed by error detection and post-selection in a repeat-until-success fashion.

Moreover, the successful realization of QEC requires low-latency integration of QPUs with GPUs, which are effective at executing a large number of identical, rela-

tively shallow computations in parallel on different input data. Such advantages are relevant to real-time decoding, as discussed 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 [41].

**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 [42], testing looks for problematic behaviors in a physical quantum computer (by running specific circuits) [43, 44], and debugging attempts to diagnose and correct problems [45]. 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 [46]. These tasks are much more complicated for quantum circuits than classical ones. For example, a quantum counterpart to the conventional equivalence-checking technique [47] 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 [48]. More-sophisticated approximate testing [49] techniques must also be developed to take the error tolerance of quantum computation into account. Finally, it is much harder to diagnose and eliminate errors [50]; 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 IV). 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, which brings its own set of challenges [51]. Optical interconnects have been proposed for providing such quantum links between distinct DRs [52–56]. 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.

Recent proposals for scalable networking of neutral-atom fault-tolerant architectures [57, 58] suggest promising prospects for distributed quantum computing. However, while the creation of atom–photon entanglement and heralded atom–atom entanglement generation based on atom–photon interface have been demonstrated in numerous experiments, deploying entanglement among remote superconducting qubits through an interface with microwave photons is a less mature but promising direction for distributed superconducting quantum computing [59].

Since quantum networking technologies are still in their infancy, in Section VI, we show how adaptive circuit knitting strategies could delay the need for optical interconnects via high-performance distributed quantum evaluation of sub-circuits 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 D on high-performance quantum–classical workload scheduling.

**Microarchitecture 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 how to converge to an effective and versatile microarchitecture. Such microarchitectures will face additional challenges of dependability; they must be designed to be reproducible, resilient, and secure [60]. Recent studies [28, 61–63] 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 [64]. This suggests a rethinking of conventional memory hierarchies (e.g., cache, L1, and L2) for quantum memory blocks. Notably, quantum random access memory (QRAM) [65] 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 [66]. Quantum LDPC codes may provide a path forward for realization of feasible quantum memory blocks.

**High-performance quantum operating system.** Given the lengthy execution time of utility-scale FTQC algorithms and the need for frequent recalibration of QPUs, a quantum operating system (OS) 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 preliminary steps toward addressing these challenges for FTQC compilers in Section III A.

An OS 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 2. An example of such a software suite is 1QBit's TopQAD (Topological Quantum Architecture Design) toolkit [62, 63, 67]. Aside from playing the role of a real-time OS for the envisioned quantum supercomputer, TopQAD is used in this paper to conduct two types of benchmarking: (a) to derive QPU and decoder performance targets, and QPU fabrication robustness targets in Section III; and (b) to perform detailed physical resource estimations for utility-scale quantum algorithms in Section IV. The compute-heavy real-time processes of the quantum OS suggest that it will ultimately be hosted on a well-integrated HPC platform communicating with the decoder and controllers in real time and refactoring the compilation/assembly of the quantum program given real-time QPU noise profiling data as depicted in Figure 2.

**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, for example, quantum machine learning (QML) applied to classical data. Such algorithms include quantum linear system solvers [68, 69], quantum clustering [70], quantum principal component analysis [71], and quantum support vector machines [72]. However, the promise for exponential quantum advantage still holds for quantum data [73] and substantial progress has been made toward overcoming the known trainability limits of quantum neural networks [74].

Moreover, processing classical data, for example, via coherent arithmetic operations, is costly for quantum computers [75, 76]. 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 [77]. In general, applications with classical inputs and outputs are fundamentally limited because quantum computers do not provide a universal advantage. To wit, there is provably no quantum advantage for standard comparison-based sorting—no quantum algorithm can solve the task asymptotically faster than conventional algorithms do [78].

Even if a given subroutine in an 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 [79]. 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 [80] with many implicit assumptions such as lack of explicit structures in the problem instances. However, the opportunity for such quadratic speedups in practice might be slim and most likely eventually washed away at scale by the huge overhead of QEC [81]. 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-inspired subroutines or quantum fluctuations. Recently, there has been tremendous interest in the family of quantum approximate optimization algorithms (QAOA) [82, 83], including numerical or theoretical studies that claim potential scaling advantages might be possible [84–87]. However, small-size effects can be misleading, and contrived nature of some benchmarking tasks misrepresent scaling advantage in practical scenarios for which highly-tuned classical heuristics are available. Historically, it has proven difficult to develop quantum heuristics that remain relevant at large scales. The search for new quantum algorithms on classical inputs remains a major research avenue.

**Validation of quantum algorithms.** There is little doubt that quantum computation can provide revolutionary advantage for quantum-mechanical problems [71, 73]. But validating 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 [88].

**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 B, we will discuss how some of the

Figure 1: 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 [26] 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, a major increase in the rate of progress over the next five years is needed. 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 multi-qubit gates that are fast and accurate enough) [18].

leading fabrication, chip manufacturing, and system integrator companies, for example, Applied Materials, Synopsys, Nvidia, and HPE, could establish a supply chain to reduce costs.

Overall, a natural question for the arrival of utility-scale quantum computing is the estimated timeline for development. This will be addressed next.

### C. 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 1, starting from 2014 when a repetition code experiment was performed at UCSB (nine qubits) [89], 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 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 timeline that has been plotted.

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—as detailed here—could hinder even staying on the current growth path.

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

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

Addressing key technical challenges listed above requires a heterogeneous full–stack quantum–classical hardware and software system architecture [9, 90]. To this end, we outline how one could adopt existing semiconductor ecosystems and conventional high-performance infrastructures to build such an architecture, which is schematically illustrated in Figure 2 (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.

Figure 3 shows a schematic architecture layout for future accelerated quantum supercomputers. Here, the key approach is a tight integration of QPUs as specialized accelerators within AI supercomputers. This approach combines the capabilities of quantum and classical computation, with AI superchips executing computationally demanding classical tasks and QPUs handling specialized quantum tasks. AI supercomputing can also play a complementary role in enhancing the capabilities of QPUs

Figure 2: 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 (QPU), probabilistic processing units (PPU), FPGA and custom-design ASICs with high-speed low-latency scale-up interconnect.

by executing, for example, QEC, state preparation, and optimization tasks. A low-latency interconnect between AI superchips and QPUs is a critical component of HPC–quantum integration in executing QEC. In addition, classical interconnects between AI superchips allow scalable parallel processing, while quantum interconnects between QPUs facilitate entanglement distribution and resource sharing for distributed quantum computing. By tightly coupling classical and quantum components through optimized interconnects, this architecture provides a scalable foundation for future quantum supercomputers.

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. To reach the 1M-qubit scale, dedicated and integrated CMOS may need to be developed to operate at cryogenic temperatures. In ad-

Figure 3: Architecture of two nodes in an accelerated quantum supercomputer. AI superchips are coupled to QPUs via low-latency quantum–classical interconnects and quantum control hardware. AI superchips communicate via classical HPC interconnects, while QPUs are connected by quantum interconnects.

dition, tight integration between control hardware and 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 (Sections III and IV and Appendices A to E) 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 and Appendices F and G): **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 logic 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.

### 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

Figure 4: 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.

*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 $T_1 \gtrsim 100\,\mu s$ 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 5. 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 4, 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 between $T_1$ 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, qubit metrology should be driven by system performance metrics based on the worst 1% of devices. For quantum computers, this is a useful design rule since an ~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 systems 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 $20\,\mu s$, it should be possible to meet this metric with reasonable $10\times$ im-



Figure 5: 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.

provements 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 100–200 μs range, with a "hero" device showing $T_1$ as long as 800 μs. 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.

**TLS modeling.** Developing sophisticated machine-learned models to optimize and characterize the structural properties of the Si-Al interface, with particular attention placed on potential oxygen incorporation, will create the best possible conditions for superconducting -based qubits. Leveraging a moment tensor potential trained on Kohn-Sham DFT data, the model can elucidate how variations in inter-facial structure directly influence the electronic behavior of aluminum, a critical factor in superconducting applications. Additionally, NEGF methods can be employed to provide a thorough analysis of electronic properties, allowing for exploration of the impact of fabrication processes on device performance. By linking structural characteristics, process variables,

| Hardware Parameter | Baseline | Target | Desired |
|---|---|---|---|
| $T_1$, $T_2$ times | 100 µs | 200 µs | 340 µs |
| $T_1$ tailedness | 71 µs | 23 µs | 23 µs |
| 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 µs | 1 µs | 1 µs |
| Measurement time | 200 ns | 100 ns | 100 ns |
| Reset time | 200 ns | 100 ns | 100 ns |
| Error suppression rate $\bar{\Lambda}$ | 2.12 | 7.5 | 13.5 |
| Error suppression rate $\Lambda$ | 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, $\Lambda$, 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. For comparison, the exponential suppression rate $\bar{\Lambda}$ following the commonly adopted model $\bar{\mu} d \bar{\Lambda}^{-(d+1)/2}$ is also provided (see Section III B for more details). 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 III D, 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.

and electronic outcomes, TLS modeling can drive innovation in the design of materials optimized for quantum computing environments.

**Fabrication.** Far from sufficient, the above academic manufacturing process shows that our models and fabrication plans for improving qubits are on track. This can be illustrated by 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-µm-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 6 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 fabrication 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 ($> 100$ µs). 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 [25, 91].

**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 information 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.



Figure 6: Atomically sharp aluminum-to-silicon substrate interface from Applied Materials' cluster tools used for qubit fabrication [92].



Figure 7: 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.

## 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 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-µm-sized junctions/mm$^2$, and critical lithography dimensions typically $\sim$1 µm. 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 diffi-

Figure 8: 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.



Figure 9: 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 10: 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-baised 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.

cult 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.

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 7. 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 92k wires can be routed across the wafer in a single wiring layer. As shown in Figure 8, transmission-line low-pass filters integrate 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 [93], and has developed new packaging solutions for Nvidia's Blackwell processors [94]. Applied Materials has also developed processes and tools for wafer-to-wafer bonding and heterogeneous integrations [95].

**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 7) 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 bottleneck is qubit measurement and readout, as typical systems require circulators and parametric amplifiers that have a volume of $\sim 1\,\mathrm{cm}^3$ or more. Qolab plans to use a readout technology that can be readily integrated into the qubit or wiring wafer, as described in Figs. 9 and 10. Readout using the Josephson photomultiplier, developed by McDermott [96], 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 where, 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.

In addition to linear tiling, serpentine pattern is possible for a more compact area. A system with 5–10 tiles can fit into a large dilution refrigerator using present-day designs. Scaling up to many more tiles would clearly require new designs for a large cryostat. Optical connects

Figure 11: Domain decomposition method (DDM) for a 1024-element antenna array.

[97] 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 11 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.

### 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 require 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 crosstalk, becomes crucial to 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 are 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, cryogenic integrated CMOS connected by a digital interface to the room-temperature control. 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.

**Room temperature and cryogenic control.** As quantum systems scale, the benefits of cryogenic control increase. Wafer-scale integration enables the connection of flex cables at 3 K 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. System complexity and size: 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, improving overall efficiency.

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 reduce 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. A power target of 1 mW per channel should allow the control to operate at 77 K. 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 10k100k 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, the predistortion capabilities of the control should be designed to address the given channels distortion and crosstalk should be addressed in the QPU and chip mount design with the control handling only 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 avoid unnecessary complexity and cost.

**Management and monitoring.** In large-scale systems, both control systems and quantum processors are prone to failure. Consequently, it is essential to incorporate mechanisms for testing functionality at both the system and component levels. Decoder statistics can offer insight into the overall operational status of the system. Additionally, benchmarks on one or few surfaces provide a means to diagnose localized failures. Benchmarks evaluating the fidelity of physical qubits can offer information on 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 such

Figure 12: (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 [98]. (c) Π-pulse amplitude and frequency 2D optimization demonstrated on a DGX Quantum system.

as telemetry and self-testing are required to detect, diagnose, and address such issues effectively. Upon detection of an error, the logic circuit orchestrator is notified, ensuring continued platform function while debugging and calibration processes resolve the faults.

To facilitate debug and control software updates while maintaining an operational system, the control platform is structured with multiple clusters. Each cluster is managed independently with the clusters operate in synchronization. This provides the isolation of individual clusters for maintenance or error handling without disrupting the broader system.

**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 [98]. 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. Near real-time 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.

**Control system integration with classical compute resources.** By their nature, quantum computing workloads require quantum–classical iterations. These include 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 the transfer of 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. The integration between the control system and classical compute resources may be characterized by the feedback loop timescale. QRT feedback loops, executed within the coherence time of the qubits, require low-latency feedback and are therefore executed on the pulse processor, a dedicated real-time processor within the control system. For multi-qubit based feedback, the control system must support scalable, low-latency data distribution. SRT calculations may be performed by the control pulse processor or by classical resources connected to the control system such as CPUs or GPUs. SRT compute resources do not need to be physically located within the same rack as the quantum controller and may be connected through low-latency optical links. NRT calculations may require significant computational power, particularly for hybrid applications such as circuit knitting, and are typically performed on the client resources, outside the control system. The quantum control may be directly connected to a cluster high-performance network that provides scalable, high-bandwidth, and low-latency connectivity to the compute servers. In addition to hardware integration, we require a unified software framework that supports co-development, compilation, and execution of the quantum and classical portions of the application.

**Efficient calibration.** Calibration impacts qubit fidelity and plays a critical role in determining the error correction code distance, which in turn directly affects the scalability and complexity of the decoding process. A significant challenge is variability in qubit fidelity; the system's overall performance is often constrained

Figure 13: 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 low-latency QEC network connects the clusters and acceleration servers that run the QEC decoders, logic 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.

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 12, 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 to repeat the calibration more frequently and to allocate a larger portion of the time to optimizing outlier qubit performance. Figure 12 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 logic circuit, orchestrate QEC decoding and surface operations, and control physical qubits. The main components of the platform illustrated in Figure 13 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 logic 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 operations. A high-performance user network is used for data and program loading as well as connectivity to HPC resources required, for example, for logic circuit compilation.

**Real-time calibration and control with AI on DGX Quantum.** Characterizing, tuning, controlling, and optimizing quantum devices is a time-consuming process requiring various physical resources and particular expertise. AI can help automate these tasks, reducing experts' time and effort while improving accuracy. Methods such as those based on neural networks and Bayesian optimization are particularly useful because they can produce accurate results from a small amount of data without relying on detailed physical models. Various AI methods have already been applied to characterize quantum devices [99–102], automate various tuning steps [103–106], optimize qubit control parameters such as gate fidelity and coherence times [99, 107–110], and enhance qubit readout [111, 112]. These methods can identify optimal settings more efficiently than traditional trial-and-error approaches. For example, neural networks can predict the ideal control parameters for qubits based on previous tuning data, while Bayesian optimization can be used to navigate the large parameter spaces involved in device calibration. The use of GPUs and AI infrastructure like those available with DGX Quantum (see Appendix F for details) is key to accelerating AI methods,

and thus to real-time calibration and control.

Another important related concept is digital twins of quantum devices. A digital quantum twin can accurately capture the stochastic and non-Markovian behavior of various types of qubits. Using this approach, a digital quantum twin is built first, then used for characterizing, tuning, controlling, and optimizing real quantum devices. Building an AI version of a digital quantum twin would require a massive number of simulations for generating synthetic data for pre-training, performing pre-training, and inference. DGX Quantum [113], which combines GPUs, quantum integration through CUDA-Q [114], and the Nvidia AI framework, could accelerate these simulations significantly. The detailed description of DGX Quantum system and CUDA-Q platform are provided in Appendices F and G respectively.

**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. A key requirement for ensuring efficient HPC or cloud integration is co-scheduling of the resources to prevent delays caused by one resource waiting for others to become available.

## 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 [115]. 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 [116, 117]. Fortunately, the *threshold theorem* guarantees that the overhead of FTQC scales as polylog($1/\epsilon$) with respect to the desired precision $\epsilon$ 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, 118] in two dimensions, such as surface codes [119] or color codes [120, 121]. Interestingly, for these codes the accuracy threshold is identical to the order–disorder phase transition critical point of certain classical Hamiltonians with quenched disorder [118, 122, 123]. 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, $\Lambda$, representing the rate of this exponential error suppression [24, 26].

In this paper, we discuss FTQC architectures based on the rotated surface code [124, 125], a $[\![d^2, 1, d]\!]$ stabilizer code with physical qubits arranged on a 2D lattice, as illustrated in Figure 14 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 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



Figure 14: 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.

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 [126], 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 $|T\rangle = \left(|0\rangle + e^{i\pi/4}|1\rangle\right)/\sqrt{2}$ 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 [127–129] 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* [125], 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, 63]. 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 [62]. 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. A quantum OS for the fault-tolerant computer must therefore perform offline and real-time QPU and decoder characteri-

zation, modelling, and performance analysis, and incorporate this information into the compilation pipeline for FTQC execution. In what follows, we discuss three main modules for such an OS: the FTQC compiler, the emulator (including a noise profiler), and the assembler, as summarized in Figure 2.

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 [62, 63, 67]. 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 (typically $d < 30$) and extrapolates logical error rates to higher distances (sometimes 100 or more, depending on the algorithm; see Table V). 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 15. 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 [130], 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 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 OS, including compilation, emulation, and assembly, deliver the exact sequence of instructions for all the stabilizer measurement rounds, logical operator measurements, and



Figure 15: Example of a logical layout of a modular fault-tolerant quantum architecture, of the sort designed by TopQAD [62, 63, 67]. In this example, two distillation levels are used, composed of four and two distillation units each from lowest to highest.

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 helps 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 15, and described in more detail in Refs. [62, 63, 67] and also in Appendix B. Figure 16 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 decoding 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 prac-

Figure 16: Decoder requirements for electronic-structure quantum simulations of the $p$-benzyne 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 stopological 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.

tical 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 [67].

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 $\Lambda$ represents the asymptotic error suppression rate when increasing the code distance by 2 (introduced in Refs. [24, 131] 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 [132] for simulating stabilizer circuits and PyMatching for decoding using the minimum-weight perfect matching (MWPM) algorithm [133]. 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 17. 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. We regress the lowest-order term of the logical error model of the surface code

$$\mu d^2 \Lambda^{-(d+1)/2} \tag{1}$$

to our numerical data, where $d$ is the code distance and $\mu$ and $\Lambda$ are fitting parameters. We refer the reader to Ref. [62] for further details on the choice of this error suppression model.

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 $\Lambda$ value is an important hardware characteristic, as it determines the rate of logical error suppression with distance [131]. We

note that the extracted $\Lambda$ values 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 $\Lambda$ for the desired model is 18(1), demonstrating an additional improvement factor of 2 in the error suppression rate as compared to the target parameter set.

We note that previous works [24, 26, 134] use the model $\bar{\mu}\bar{\Lambda}^{-(d+1)/2}$ 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

$$\bar{\mu}d\bar{\Lambda}^{-(d+1)/2}, \tag{2}$$

which has a linear coefficient $d$, as opposed to $d^2$ as in our model. In Figure 17, we show the discrepancies between the two choices in predicting logical error rates, especially at greater distances and with less-performant hardware specifications. The values of all fitting parameters for both models and our three hardware parameters are summarized in Table II.



Figure 17: Prediction of high-distance error rates from numerical simulations at lower distances. The numerical dataset is plotted alongside extrapolations derived from the two-parameter fits to the models Equation (1) (shown using solid lines) and Equation (2) (shown using dashed lines). The dataset is obtained using Clifford simulations based on the noise model described in this section. The fitting parameters are provided in Table II. 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

It is often unclear *a priori* which types of noise and errors have the most significant impact on the performance of FTQC, and which hardware parameters are the most

|  | Baseline | Target | Desired |
|---|---|---|---|
| $\Lambda$ | 2.34(1) | 9.3(3) | 18(1) |
| $\mu$ | 0.0038(2) | 0.019(4) | 0.04(1) |
| $\bar{\Lambda}$ | 2.119(6) | 7.5(1) | 13.5(3) |
| $\bar{\mu}$ | 0.0259(8) | 0.055(6) | 0.082(9) |

Table II: Fitting parameters obtained for the models displayed in Equation (1) and Equation (2) when regressed to the numerical data of our three hardware parameter sets, specified as baseline, target, and desired.

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 study the performance sensitivity of FTQC to specific hardware characteristics, as specified in Table I, by assessing the logical error suppression factor $\Lambda$ as a function of individual hardware parameters. We analyze the following categories of hardware parameter improvements in the operations that implement 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 $\Lambda$ when each of these parameter sets are improved separately, while keeping the others constant, as well as when all hardware parameters are improved simultaneously. Figure 18 illustrates our findings. 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 $\Lambda$ values. The $\Lambda$ value increase resulting from improving all parameters simultaneously is higher than the sum of individual increments of $\Lambda$. Our findings suggest that quantum gate fidelity improvements are more important than SPAM and idling qubit error rates for achieving greater logical performance.

## 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 analyze 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

Figure 18: Sensitivity of logical infidelity to hardware improvements. The extracted fit parameter $\Lambda$, 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.

obtained for the ibm_torino quantum processor [135]. In particular, we focus on the distributions of the $T_1$ times, as well as single-qubit, two-qubit, and readout errors. In Figure 19 we plot the cumulative distribution functions (CDF) of this data. Physically, we expect some correlations between these distributions, for example, a 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 19: 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 [135]. Dashed vertical lines indicate the mean values. Legends indicate standard deviations.

To capture these correlations, we employ a random-forest model [136], 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 20 shows that our random-forest models adequately estimate the gate and readout errors.



Figure 20: 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 $\rho$ are shown at the top of each figure.

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 $\sigma$ of the distribution. Therefore, we construct several synthetic $T_1$ distributions with varying values of $\sigma$, by rescaling the IBM $T_1$ distribution,

$$T_1 \rightarrow \mu + a(T_1 - \mu), \tag{3}$$

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



Figure 21: (a) Transformed $T_1$ distributions with different standard deviation, $\sigma$. 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 21(a), we show the CDFs of three distributions generated by applying the transformation (3).

| 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 III: Gate times of the `ibm_torino` QPU.

The three values of $\sigma$ 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:

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 with assigned gate fidelities, and assuming gate times from the ibm_torino device data (see Table III), to determine the logical error rate for the code.

The output distributions of the logical error rates are shown in Figure 21(b). We observe that higher values for $\sigma$ 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 FTQC is the preparation of high-fidelity logical magic states. These states are produced by first employing a magic state preparation protocol [127–129], 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 greater distances and fed to magic state distillation units to prepare a lower number of higher-fidelity magic states. These magic state distillation units are able to perform only if they are fed logical magic states of sufficient fidelity. It has been estimated that the 15-to-1 distillation units have an acceptance probability

$$1 - 15P_{\text{magic}} - 356P_{\text{Cliff}}, \tag{4}$$

where $P_{\text{magic}}$ and $P_{\text{Cliff}}$ are the logical error rates of input logical magic states and Clifford operations, respectively [61]. Hence, we need to produce logical magic states with error probability

$$P_{\text{magic}} < \frac{1 - 356P_{\text{Cliff}}}{15}. \tag{5}$$

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 [128] 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 code distance $d$. The simulation results reported in Figure 22 show that the error rates increase with $d$, 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 (5) and draw the respective threshold curves for each of the hardware parameters shown in Figure 22. We observe that both the target and desired parameter sets are significantly below their respective 15-to-1 distillation thresholds, unlike the baseline set.

A more recent protocol [129] 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 22. We observe that all three parameter sets are significantly below the 15-to-1 distillation threshold for this protocol and for this reason 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 Ref. [137], space-like errors are exponentially suppressed by increasing the code distance of the logical qubits. Similarly, time-like errors are exponentially reduced by increasing the number of stabilizer measurement rounds during the merge operation

(see Figure 23). 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 $P_\psi$ 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 $F_p = \frac{(D+1)(1-P_a)-1}{D} = 1 - 3/2 P_a$, where $D = 2$ is the dimension of the Hilbert space of the single qubit being teleported [138, 139]. Since accessing the $Y$ operator of the surface code is cumbersome [140], we teleport the logical states $|\psi_L\rangle \in \{|0_L\rangle, |+_L\rangle\}$ using an $XX$ (rough) merge, as illustrated in Figure 23. 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 illustrated in Figures 23 and 24(a) and outlined as follows:

1. Preparation (Figure 23, left side): We begin by preparing the logical source state $|\psi_L\rangle \in$



Figure 22: Performance of the magic state preparation protocols Gidney2023 [128] and Gidney2024 [129] 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 (5) for the baseline, target, and desired parameter set, respectively, where the difference is due to the fact that $P_{\text{Cliff}}$, estimated from the memory protocol logical error rate, depends on hardware noise. We observe that the baseline set is below the respective 15-to-1 distillation threshold only for the Gidney2024 protocol; it is above the threshold for the Gidney2023 protocol.

$\{|0_L\rangle, |+_L\rangle\}$ and the target patch in the state $|0_L\rangle$. In this pre-merging step, these states are stabilized for $r_{\text{pm}}$ rounds.

2. Merging (Figure 23, middle): After the $r_{\text{pm}}$ rounds, the bus data qubits are respectively initialized in the physical $|0\rangle$ state, and the rough edges of the two surfaces are merged. Then, the entire surface is stabilized for $r_{\text{m}}$ rounds to determine (a possibly erroneous) measurement of the $X \otimes X$ observable with an outcome $m_1$.

3. Splitting (Figure 23, right side): After the $r_{\text{pm}} + r_{\text{m}}$ rounds, the bus data qubits are respectively measured in the $Z$ basis (splitting) and the remaining patches are stabilized for $r_s$ rounds. Note that this also results in a perfect round of syndrome measurements on the bus patches.

4. Projection (Figure 23, right side): After the $r_{\text{pm}} + r_{\text{m}} + r_s$ rounds, the source data qubits are respectively measured in the $Z$-basis, to determine the (possibly erroneous) value of an outcome $m_2$.

5. Recovery (Figure 23, right side): 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 mentioned teleportation protocol for varying spatial code distances $d$, temporal code distances $r_{\text{m}}$, and bus widths $b$. We then regress the following predictive models from the obtained numerical results of $P_0$ and $P_+$ values to predict the fidelity of the protocol at large distances [62, 137]:

$$P_0 = \mu_X (2d + b) r_{\text{m}} \Lambda_X^{-(d+1)/2}, \tag{6}$$

$$P_+ = \mu_Z d \Lambda_Z^{-(d+1)/2} + \mu_T db \Lambda_T^{-(r_{\text{m}}+1)/2}. \tag{7}$$

In our simulations, the number of pre-merge stabilization rounds is fixed at $r_{\text{pm}} = 1$, during which the two logical states are prepared. For the remaining rounds, we incorporate the circuit-level noise model detailed in Appendix E using the baseline noise parameters. We also set $r_s = 0$. For our benchmarking 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 at the target patch.

Figure 25(a) shows the logical error rates in 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 a corresponding bus width $b = 3d$ and a temporal distance $r_{\text{m}} \in \{d, 3d\}$. We highlight two observations from Figure 25(a):

- The error rates are suppressed as a function of code distance for both states. This indicates that the noise parameters are below the threshold.

Figure 23: Logical teleportation via lattice surgery for a rotated surface code of distance $d = 3$ and bus width $b = 1$. Illustrated are the states prior to merging (left), during merging (middle), and after splitting (right). The wire diagram of the teleportation 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 of $m_1$ and $m_2$. We do not simulate the circuity of the recovery operations $Z_L$ and $X_L$; instead, we measure all qubits at the dashed line to determine the performance of the teleportation.



Figure 24: Space–time diagrams of 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 28, and the temporal dimension is divided into $r_{pm}$ pre-merge rounds and $r_m$ merge rounds. (b) 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 (represented by the left code patch) and data qubits (represented by the right code patch). The magic state may incur a delay $\tau_b$ in the buffer before the surgery is performed. The magic state and the bus are measured after the merge operation, but target patches must await the decoder decisions, available after a decoder delay time $\tau_d$. These idling patches must be protected by a quantum memory protocol using further stabilization rounds during this period; therefore, the accumulation of further errors is inevitable and must be taken into account by the compiler, FTQC emulator, and resource estimators.

- Increasing $r_m$ decreases the teleportation fidelity of $|0_L\rangle$ while increasing the fidelity of teleporting $|+_L\rangle$ (see also Figure 25(d)).

Figure 25(b) shows the fitting for the $X$- and $Z$-type terms of Equations (6) and (7). This model predicts the $X$- and $Z$-type errors better in the high-$r_m$ regime (hence our choice of the $r_m = 3d$ data). Similarly, Figure 25(c) shows an estimate of the time-like error suppression term in Equation (7) in the high-distance regime, yielding the values $\mu_T \approx 0.0273(6)$ 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, Figure 25(d) shows the optimal number of merge stabilization rounds $r_m$ for a given distance $d$. Here, we choose $d = 7$ and consider 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 values of $r_m > 7$, highlighting the fact that the optimal number of QEC rounds for a lattice surgery operation with code distance $d$ may actually deviate from the commonly assumed value $d$.

For the teleportation of magic states in the core processor, the magic state has resided in the buffer for some average expected buffer delay time $\tau_b$, 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 becomes available. We denote this delay by $\tau_d$. Inclusion of the buffer and decoder delays and assuming an average rate for all types of surgeries results in the model

$$\mu\left[d(2r + \tau_b + \tau_d + 1) + br\right]\Lambda^{-(d+1)/2} + \mu_T db\Lambda_T^{-(r+1)/2}, \quad (8)$$

which still distinguishes time-like and space-like errors but ignores the type of surgery (see Figure 24(b)).

Figure 25: (a) Error suppression as a function of code distance $d$ for a bus width $b = 3d$ and for $r_m \in \{d, 3d\}$ merge rounds, for the teleportation of $|0_L\rangle$ ($P_0$) and $|+_L\rangle$ ($P_+$). (b) Exponential fit for $P_0(r_m = 3d)$ and $P_+(r_m = 3d)$ using 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) Exponential suppression of $P_+$ as a function of the number of rounds for $r_m < d$, 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 $|0_L\rangle$ and $|+_L\rangle$ as a function of the merge stabilization rounds (temporal distance) for a code distance $d = 7$ over a bus width $b \in \{d, 3d\}$ and the corresponding average state infidelities calculated using $\frac{2}{3}(P_0 + P_+)$.

## 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 is not known yet what algorithms are most effective at decoding; therefore, there is a systems engineering trade-off between 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 [108]. 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 ($L^0$ and $L^1$ in Figure 26(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, as shown in Figure 26(b). 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 the delay remains within tens of microseconds and on average the decoder result is ready on time [141]. Therefore, we target an end-to-end average decoding latency shorter than $d$ QEC cycles, which for superconducting qubits implies a target latency of approximately 10µs . 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. For 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 instructions. 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 process, which includes multiple concurrent decoding tasks [141], should be designed to minimize the result 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 to not limit performance, the average decoder throughput should exceed the syndrome generation rate. In addition, the average decoder 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 13 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 surfaces, 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 logic 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 of-

fload capabilities. In large-scale systems, multiple acceleration servers may operate in parallel, running multiple 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 10µs, which require the design of a specialized communication protocol for QEC.

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

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 future DGX Quantum platform, based on the Blackwell architecture B200 GPUs, utilizes PCIe 6.0, which provides up to 128 GB/s bidirectional bandwidth in an x16 configuration. The PCIe 6.0 specification incorporates Forward Error Correction (FEC) to maintain data integrity and limit additional latency to under two nanoseconds, ensuring high-speed data transfers with low latency. 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. B200 GPU achieves up to 18 petaflops (PFLOPS) for sparse operations. This is made possible through advanced sparsity techniques and architectural enhancements, doubling the effective performance in scenarios where sparsity can be exploited effectively, which is often the case for quantum circuit simulations given the sparsity of data.

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 is desired in the early stages of FTQC.

Preliminary evaluation of a software-based implementation of the Fusion Blossom algorithm with batch decoding on a DGX Quantum server demonstrated that a

Figure 26: Example of non-Clifford computation with surface codes adapted from Ref. [108]. (a) Example of a logic circuit containing two non-Clifford gates. (b) Fault-tolerant logic 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) 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 ($L_{\text{th}}$).

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.

The DGX Quantum platform is particularly effective for QEC schemes that benefit from local decoding, as these can be efficiently accelerated by GPUs. Local decoders, such as Steiner tree-based decoders, are prime examples of algorithms that perform well in GPU-accelerated environments due to their parallelizable structure. In addition to standard local decoding, there are more-complex classes of quantum codes, such as entanglement-assisted quantum error-correcting codes (EAQECC) and entanglement-assisted quantum low-density parity-check (EA-QLDPC) codes, which offer enhanced error-correction capabilities by leveraging entanglement [142, 143]. Some of these codes, however, have varying degrees of GPU acceleration potential, depending on the complexity and structure of their decoding algorithms. Some EA-QLDPC codes, particularly those with irregular or non-local error syndromes, require more-sophisticated scheduling but can still benefit from GPU acceleration with optimized parallelization strategies. Recent research efforts have focused on discovering new classes of quantum codes that not only offer high error-correction rates but are also capable of full utilization of GPUs.

AI-assisted decoders and quantum algorithms can also significantly improve real-time decoding performance by leveraging rich AI infrastructure and GPU acceleration [113]. A great example of AI's application to QEC is Google's AlphaQubit approach, which leverages machine learning to enhance the decoding process for QEC codes [144]. AlphaQubit uses reinforcement learning and neural networks to identify optimal error-correction strategies by learning from a large amount of synthetic and experimental data.

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 the 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 $d$ opti-

Figure 28: 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 $b \times d$.

Figure 27: Example of an embedded multi-DR architecture for executing a quantum circuit associated with electronic-structure quantum simulation with a target error of 0.01 mHa based on qubitization of the $p$-benzyne 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.

cal interconnects between nearest-neighbour DRs, which is a demanding requirement [52–56].

**Assembling large FTQC programs among multiple DRs.** Embedding FTQC architectures across multiple DRs involves solving a complex embedding problem to determine the connectivities between DRs, teleporation 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 remains 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 27 illustrates an example of an embedded multi-DR architecture designed for executing the $p$-benzyne 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 28. In Figs. 29(a) and (b), we show that using weaker CNOT gates 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 four cuts within the bus. However, much weaker interconnects can be problematic as shown in Figs. 29(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. Our numerical results are consistent with an analogous analysis performed for neutral-atom computers in [58].

We note that the approach based on multi-DR lattice surgeries described in this section competes with the entanglement-distillation based quantum networking approach proposed in Ref. [57]. In this approach, time-multiplexed remote physical Bell pairs are used to encode noisy logical Bell pairs by state injection, followed by entanglement distillation using logical operations on the code blocks to reach a desired error rate for the teleported state in the target QPU. However, the entanglement distillation units create a resource trade-off between the number of optical interconnects and the number of qubits used to complete the teleportation scheme in the target QPU. The analysis of such trade-offs is left for future resource estimations.

## 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 [62, 63, 67] and Azure-QRE [145].

Figure 29: (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 $n_{\text{cuts}} = 4$. The obtained $\mu$ and $\Lambda$ values are close to that of Figure 25(b). (c)–(d) Logical infidelity of teleporting $|0_L\rangle$ states and $|+_L\rangle$ states, respectively, as a function of the infidelity of the weak $CX$ gates in the cuts.

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

One of the most promising applications of quantum computing is sol 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 [146], 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*-benzyne molecule (*p*-benzyne), which has the molecular formula $C_6H_4$. Its energetically lowest configuration 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 [147, 148]. 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, 149–151].

For our QRE studies, we first generate the logical quantum circuits associated with electronic-structure simulations of the *p*-benzyne 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 [152, 153]. For the *p*-benzyne 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 $\{\phi_\alpha(\boldsymbol{x})\}$ 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 ($\hat{a}_\alpha$ and $\hat{a}_\alpha^\dagger$) 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. 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 [154] or the Bravyi–Kitaev [155] transformations. To derive the model Hamiltonians for the various active spaces associated with $p$-benzyne, we used Tangelo [156], 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 [157] of Ref. [150].

The standard QPE algorithm [153] 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 $\epsilon$ 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 [158]. Indeed, most of the recent QRE studies on electronic-structure quantum simulations rely on the qubitization framework (see, e.g., Refs. [61, 150, 159–165]). This approach uses a new operation called *qubiterate* that is akin to the quantum walk operator $e^{i\arccos(H/\lambda)}$ (where $\lambda$ 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. [166, 167]. 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. [159], the double low-rank factorization algorithm of von Burg et al. [160], and the tensor hypercontraction algorithm of

Lee et al. [150], 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 $\lambda$ is the 1-norm of Hamiltonian coefficients which typically has a scaling between $\mathcal{O}(N)$ and $\mathcal{O}(N^3)$, and $\epsilon$ specifies the target error in ground-state energy estimation.

## B. Quantum resource estimation 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. [160] and further analyzed in Ref. [150]. 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 logic 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. [160]. 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 challenge is problem-dependent but 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 [88] 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. [77])

$$\mathcal{O}\left(\frac{g(\epsilon_{\mathrm{QPE}})\Omega}{\epsilon_{\mathrm{QPE}}}\|W'(H)\|^{-1}\right). \qquad (9)$$

Here, $\epsilon_{\mathrm{QPE}}$ 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 $\tau$, or $W(H) = e^{i\arccos(H/\lambda)}$ in the case of qubitization); $\Omega$ 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 $g(\epsilon_{\mathrm{QPE}})$ 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 $\epsilon_{\mathrm{QPE}}$. 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 $g(\epsilon_{\mathrm{QPE}}) = \mathcal{O}(1)$. Hence, due to $\|W'(H)\|^{-1} \leq \lambda$, the overall cost of QPE in qubitization-based approaches becomes $\mathcal{O}(\Omega\lambda/\epsilon_{\mathrm{QPE}})$. Various versions of QPE have been analyzed in the literature, aiming to reduce its cost. For example, the standard QPE algorithm [153] allows the estimation of eigenvalues within a target error $\epsilon_{\mathrm{QPE}}$ with probability at least $1/2$ using $\lceil 16\pi/\epsilon_{\mathrm{QPE}}\rceil$ applications of the unitary $\exp(-iHt)$. However, more optimized QPE strategies can achieve the multiplying factor (see Refs. [35, 77, 150])

$$M := \lceil \pi/(2\epsilon_{\mathrm{QPE}})\rceil. \qquad (10)$$

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 $\Omega$ comprises the costs of the LCU oracles SELECT and PREPARE.

In Tables IV and V, 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 $N_{\mathrm{orb}}$, 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. [168]). 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 $\epsilon = 1.6$ 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 $p$-benzyne, 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 (10). For the overall error budget of $\epsilon = 0.1$ mHa, we obtain a value of $\epsilon_{\mathrm{QPE}} = 0.065$ mHa as an optimal choice (see Appendix A 2), yielding $M = 24{,}167$; for the overall target error $\epsilon = 1.6$ mHa, we use $\epsilon_{\mathrm{QPE}} = 1.04$ mHa, yielding $M = 1511$. 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 IV, 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 $\mathcal{O}(\log^c(1/\varepsilon))$, with the exponent $c > 3$, and the software package gridsynth [169] based on the algorithm by Ross and Selinger [170, 171] achieving $T$-gate counts that are typically on the order of $4\log_2(1/\varepsilon) + \mathcal{O}(\log(\log(1/\varepsilon)))$, for a given allowable per-gate synthesis error $\varepsilon$. We use the latter method in our QRE studies due to its superior scaling.

As explained in Refs. [28, 62, 63], Clifford operations can be efficiently commuted to the end of the logic 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 V, 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 with baseline, target, or desired parameter values.

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

| Molecule Specification | | Logical Resources | | | |
| --- | --- | --- | --- | --- | --- |
| | | $\epsilon = 1.6$ mHa | | $\epsilon = 0.1$ mHa | |
| Active space | Number of orbitals, $N_{\text{orb}}$ | # Qubits | # $T$ gates | # Qubits | # $T$ gates |
| **Rigor. Trotter** | | | | | |
| $p$-benzyne, HL$\pm 2$ | 6 | 12 | $4.5 \times 10^9$ | 12 | $4.1 \times 10^{11}$ |
| $p$-benzyne, HL$\pm 6$ | 14 | 28 | $4.6 \times 10^{11}$ | 28 | $3.8 \times 10^{13}$ |
| $p$-benzyne, HL$\pm 8$ | 18 | 36 | $2.0 \times 10^{12}$ | 36 | $1.6 \times 10^{14}$ |
| $p$-benzyne, HL$\pm 12$ | 26 | 52 | $1.7 \times 10^{13}$ | 52 | $1.4 \times 10^{15}$ |
| **Empir. Trotter** | | | | | |
| $p$-benzyne, HL$\pm 2$ | 6 | 12 | $2.7 \times 10^8$ | 12 | $2.5 \times 10^{10}$ |
| $p$-benzyne, HL$\pm 6$ | 14 | 28 | $3.5 \times 10^9$ | 28 | $4.0 \times 10^{11}$ |
| $p$-benzyne, HL$\pm 8$ | 18 | 36 | $1.1 \times 10^{10}$ | 36 | $8.0 \times 10^{11}$ |
| $p$-benzyne, HL$\pm 12$ | 26 | 52 | $2.7 \times 10^{10}$ | 52 | $2.3 \times 10^{12}$ |
| **DF Qubitization** | | | | | |
| $p$-benzyne, HL$\pm 2$ | 6 | 291 | $4.9 \times 10^6$ | 341 | $9.0 \times 10^8$ |
| $p$-benzyne, HL$\pm 8$ | 18 | 502 | $2.3 \times 10^9$ | 568 | $4.6 \times 10^{10}$ |
| $p$-benzyne, HL$\pm 12$ | 26 | 668 | $9.0 \times 10^9$ | 748 | $1.8 \times 10^{11}$ |
| FeMoco [36] | 76 | 1789 | $7.7 \times 10^{11}$ | 1972 | $1.4 \times 10^{13}$ |

Table IV: 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 1.6 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 $N_{\text{orb}}$. 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. [160]. The reported $T$-gate counts are obtained after circuit synthesis over the Clifford$+T$ gate set.

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 IV and Table V, 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 30, 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 [172], which were calculated by extrapolating the results of recent classical calculations [163, 173]. 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 up to 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

| | Target error $\epsilon$ | $N_{orb}$ | Baseline Parameter Set | | | Target Parameter Set | | | Desired Parameter Set | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | # Phys. qubits | Phys. time | QEC code distances | # Phys. qubits | Phys. time | QEC code distances | # Phys. qubits | Phys. time | QEC code distances |
| **Rigor. Trotter** | 1.6 mHa | 6 | $3.5\times10^7$ | 2.5 days | 15, 33, 75 \| 87 | $1.6\times10^6$ | 13.7 hours | 11, 29 \| 31 | $4.4\times10^5$ | 11.1 hours | 21 \| 25 |
| | | 14 | $3.7\times10^7$ | 282.8 days | 17, 37, 87 \| 97 | $2.2\times10^6$ | 64.9 days | 13, 33 \| 35 | $7.6\times10^5$ | 50.1 days | 25 \| 27 |
| | | 18 | $3.9\times10^7$ | 3.5 years | 17, 39, 91 \| 101 | $2.4\times10^6$ | 293.7 days | 13, 35 \| 37 | $9.2\times10^5$ | 230.2 days | 27 \| 29 |
| | | 26 | $4.5\times10^7$ | 33.5 years | 17, 43, 97 \| 111 | $3.2\times10^6$ | 7.5 years | 15, 37 \| 39 | $1.1\times10^6$ | 6.0 years | 29 \| 31 |
| | 0.1 mHa | 6 | $3.5\times10^7$ | 248.1 days | 17, 37, 87 \| 95 | $2.3\times10^6$ | 58.2 days | 13, 33 \| 35 | $8.4\times10^5$ | 44.9 days | 25 \| 27 |
| | | 14 | $4.7\times10^7$ | 76.6 years | 21, 41, 97 \| 117 | $3.4\times10^6$ | 16.3 years | 15, 37 \| 39 | $1.8\times10^6$ | 12.9 years | 11, 29 \| 31 |
| | | 18 | $4.3\times10^7$ | 323.2 years | 19, 41, 103 \| 117 | $3.3\times10^6$ | 72.1 years | 15, 39 \| 41 | $1.9\times10^6$ | 58.0 years | 11, 31 \| 33 |
| | | 26 | $4.7\times10^7$ | 2839.9 years | 19, 45, 107 \| 119 | $3.4\times10^6$ | 683.4 years | 15, 41 \| 45 | $2.6\times10^6$ | 501.2 years | 13, 31 \| 33 |
| **Empir. Trotter** | 1.6 mHa | 6 | $1.1\times10^7$ | 3.2 hours | 29, 69 \| 77 | $6.7\times10^5$ | 46.5 minutes | 25 \| 29 | $3.7\times10^5$ | 36.9 minutes | 19 \| 23 |
| | | 14 | $4.1\times10^7$ | 2.1 days | 19, 31, 75 \| 95 | $1.6\times10^6$ | 10.7 hours | 11, 29 \| 31 | $4.9\times10^5$ | 8.6 hours | 21 \| 25 |
| | | 18 | $3.1\times10^7$ | 6.4 days | 15, 33, 79 \| 95 | $1.5\times10^6$ | 1.5 days | 11, 29 \| 35 | $6.3\times10^5$ | 1.1 days | 23 \| 25 |
| | | 26 | $3.7\times10^7$ | 16.0 days | 15, 35, 81 \| 93 | $2.0\times10^6$ | 3.8 days | 13, 29 \| 33 | $6.7\times10^5$ | 2.7 days | 23 \| 25 |
| | 0.1 mHa | 6 | $3.4\times10^7$ | 14.4 days | 15, 35, 81 \| 89 | $2.1\times10^6$ | 3.4 days | 13, 29 \| 33 | $6.6\times10^5$ | 2.6 days | 23 \| 25 |
| | | 14 | $3.5\times10^7$ | 239.2 days | 17, 37, 87 \| 95 | $2.4\times10^6$ | 56.1 days | 13, 33 \| 35 | $8.9\times10^5$ | 43.3 days | 25 \| 27 |
| | | 18 | $3.6\times10^7$ | 1.4 years | 17, 37, 89 \| 99 | $2.3\times10^6$ | 120.3 days | 13, 33 \| 37 | $8.9\times10^5$ | 94.3 days | 25 \| 29 |
| | | 26 | $3.8\times10^7$ | 4.2 years | 17, 39, 91 \| 103 | $2.7\times10^6$ | 346.7 days | 13, 35 \| 37 | $1.1\times10^6$ | 271.7 days | 27 \| 29 |
| **DF Qubitization** | 1.6 mHa | 6 | $1.4\times10^7$ | 3.3 minutes | 25, 59 \| 73 | $1.3\times10^6$ | 1.2 minutes | 23 \| 27 | $8.2\times10^5$ | 35.8 seconds | 17 \| 21 |
| | | 18 | $4.8\times10^7$ | 1.4 days | 15, 31, 75 \| 93 | $3.6\times10^6$ | 7.3 hours | 11, 27 \| 33 | $1.7\times10^6$ | 5.6 hours | 21 \| 25 |
| | | 26 | $5.5\times10^7$ | 5.4 days | 15, 33, 79 \| 95 | $4.7\times10^6$ | 1.3 days | 11, 29 \| 35 | $2.6\times10^6$ | 23.6 hours | 23 \| 27 |
| | | 76 | $1.2\times10^8$ | 1.4 years | 17, 37, 89 \| 107 | $1.3\times10^7$ | 121.8 days | 13, 33 \| 39 | $7.7\times10^6$ | 96.8 days | 25 \| 31 |
| | 0.1 mHa | 6 | $2.3\times10^7$ | 12.6 hours | 29, 77 \| 91 | $3.0\times10^6$ | 2.7 hours | 11, 27 \| 31 | $1.4\times10^6$ | 2.2 hours | 21 \| 25 |
| | | 18 | $5.8\times10^7$ | 30.8 days | 15, 35, 91 \| 105 | $5.1\times10^6$ | 6.5 days | 13, 31 \| 35 | $2.6\times10^6$ | 5.4 days | 23 \| 29 |
| | | 26 | $7.9\times10^7$ | 132.1 days | 19, 35, 85 \| 115 | $6.8\times10^6$ | 28.5 days | 13, 31 \| 39 | $3.4\times10^6$ | 21.2 days | 25 \| 29 |
| | | 76 | $1.5\times10^8$ | 28.5 years | 17, 41, 99 \| 119 | $1.8\times10^7$ | 6.5 years | 15, 35 \| 43 | $9.8\times10^6$ | 5.0 years | 29 \| 33 |

Table V: Physical resource estimates generated using the TopQAD toolkit [67] for implementing the QPE algorithm on electronic-structure quantum circuits associated with the $p$-benzyne and FeMoco molecules, for two precisions in energy estimation: qualitatively accurate computation within a target error 1.6 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 IV. 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 $N_{orb} = 6, 14, 18, 26$ correspond to active space selections HL±2, 6, 8, 12 (using HL±$n$ to denote "HOMO−$n$ and LUMO+$n$"; see Appendix A 1 for an explanation of these terms) for $p$-benzyne using the `6-31G` basis to represent the fermionic orbitals; the data for $N_{orb} = 76$ 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, 87 | 97] means that the code distances $d = 17, 37$, and 87 are required for the first, second, and third magic state distillation levels, respectively, while the QEC code distance $d = 97$ is needed to encode the logical qubits of the core processor. These choices are determined by the architecture's assembler [67] based on optimizations of the various trade-offs between the space and time costs proposed in Ref. [62]. Physical resources are reported for the same three quantum algorithmic approaches as in Table IV. The associated resource requirements are reported for three hardware specifications, namely, baseline, target, and desired hardware, as summarized in Table I. The results of this table are plotted in Figure 30.

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 [172] run with parallel processing on HPC units can be significantly faster than quantum simulations for molecular active spaces involving up to $N_{orb} \approx 50$ spatial orbitals, as can also be observed in Figure 30(a). The

most recent cutting-edge petaFLOPS performance results for a hybrid CPU–multi-GPU parallel implementation of the spin-adapted ab initio DMRG method on the state-of-the-art Nvidia DGX-H100 architecture suggest that heuristic classical simulations of molecular systems are tractable even for sizes well above 75 spatial orbitals [174]. 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 large bond dimensions, and thus intractable runtimes [35, 168]. While there is no such sharp transition line between what is classically tractable and classically intractable (which is highly dependent on the

Figure 30: Physical resource estimates for electronic-structure quantum computations for two molecules, $p$-benzyne and FeMoco, and for two precisions in energy estimation: (a) qualitatively accurate simulation within a target error 1.6 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 $N_{\mathrm{orb}}$. The data for $N_{\mathrm{orb}} = 6, 14, 18, 26$ correspond to the active-space specifications 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) for $p$-benzyne using the `6-31G` basis set to represent the fermionic orbitals; the data for $N_{\mathrm{orb}} = 76$ 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. [160]. Furthermore, the associated resource requirements are reported for three hardware specifications: baseline, target, and desired hardware, as summarized in Table I. For comparison, for energy estimations within the target error 1.6 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 [163, 173].

extent of quantum correlations in the studied systems), a quantum advantage gradually appears for spatial orbital numbers beyond $N_{\mathrm{orb}} \approx 50$. 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.

The FeMoco molecule has been representatively used as an important utility-scale use case for fault-tolerant quantum simulations in quantum chemistry in several recent QRE studies [35, 149–151]. Reiher et al. [35] pre-

sented the first thorough QRE analysis for FeMoco for an active space involving 54 spatial orbitals, reporting a $T$-gate count on the order of $10^{14}$ for qualitatively accurate computations (1.0 mHa), and on the order of $10^{15}$ for quantitatively accurate computations (0.1 mHa), for quantum simulations based on the Trotter approach. Assuming that a logical $T$ gate can be implemented within a time of $\sim 100$ ns, which is a highly optimistic assumption, they conclude that runtimes on the order of several months to several years would be required to estimate ground-state energies of FeMoco within a precision of 1.0 mHa or 0.1 mHa. However, the active space analyzed by Reiher et al. (involving 54 spatial orbitals) was

later pointed out not to be a suitable model for representing the ground state of FeMoco, because it does not correctly capture the open-shell character of the underlying molecular system, and a more appropriate active space (involving 76 spatial orbitals) was proposed by Li et al. [36]. Hence, our QRE study pertains to this much larger FeMoco active space. The reported runtimes for the DF qubitization algorithm are on the order of four months (for an accuracy of 1.6 mHa) to a few years (for 0.1 mHa), when using the target hardware specifications. Yet, more-optimistic logical resource counts and associated physical runtimes have been recently reported for quantum simulations of the Li et al. FeMoco. In particular, Lee et al. [150] propose more-efficient electronic-structure quantum simulations based on qubitization through tensor hypercontraction (here referred to as "THC qubitization") and demonstrate that the Li et al. FeMoco can be simulated within a qualitative chemical accuracy of 1.6 mHa with a Toffoli count of $3.2 \times 10^{10}$, implemented with four million physical qubits and a runtime of only four days, using an FTQC architecture based on CCZ magic state factories and a self-correcting CCZ (AutoCCZ) state consumption [175]. We note that the runtime for FeMoco simulations could possibly be reduced even further through the recently proposed symmetry-compressed double factorization approach [176]. Such recent developments in advancing quantum simulation algorithms and in architectural design considerations are a promising avenue for further reducing the expected runtime for estimating the ground-state energy of FeMoco.

It is insightful to understand the discrepancy between the runtimes reported here for the DF qubitization algorithm and the four days reported for THC qubitization in Ref. [150], for quantum simulations of the Li et al. FeMoco within the chemical accuracy 1.6 mHa. For instance, the 122 days and 97 days that we report as runtimes for the target and desired hardware specifications, respectively (see Table V), are longer by the factors of approximately 30 and 24 than the four days reported in Ref. [150]. We note that, on the logical level, the $T$-gate count reported here for DF qubitization (see Table IV) is higher by only a factor of 6 than the $T$ gate count resulting from converting the Toffoli gate count reported for THC qubitization in Table III of Ref. [150], assuming that at least four $T$ gates are required for decomposing each Toffoli gate into gates from the Clifford+$T$ gate set; importantly, apart from using different representations of the Hamiltonian in the two algorithms (both based on qubitization), the use of different criteria for determining the truncation thresholds in the double factorization and tensor hypercontraction approaches can significantly contribute to the difference in the reported $T$-gate counts. Thus, the discrepancy between the reported runtimes is not entirely due to running a different quantum simulation algorithm (DF qubitization vs. THC qubitization), but also to differing assumptions on the physical and architectural levels. The resource requirements reported in Ref. [150] are based on using AutoCCZ states (and the associated CCZ factories) to implement the Toffoli gates, while the resource estimates reported here are based on $T$ magic states (and the associated factories) to implement the $\pi/8$ rotations. We expect that supporting CCZ factories in our architecture will allow for up to a four times faster consumption of magic states depending on the input logic circuit. Together, the factor of 4 (on the architectural level) and factor of 6 (on the logical level) result in a factor of 24 for the expected difference between the runtime for DF qubitization reported in this work and the runtime for THC qubitization in [150], which explains the difference in 122 days (or 97 days) versus four days. Further differences arise due to using different hardware specifications. The physical noise model assumed in Ref. [150] results in the values $\Lambda \approx 6.1$ and $\bar{\Lambda} \approx 5.0$ for the error suppression rates associated with the error suppression models discussed in Section III B. In addition, different physical space–time trade-offs can even be obtained for the same logic circuit and the same noise model, as is illustrated in Figure 31 for the circuit associated with quantum simulation of the Li et al. FeMoco based on THC qubitization, given the architecture discussed in this paper. Such space–time trade-off considerations are discussed in more detail in Ref. [62].

## 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 outright 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-performance computing (quantum–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.

Figure 31: Space–time trade-offs for implementing a quantum circuit associated with quantum simulations of the Li et al. active space [36] for FeMoco based on the THC qubitization algorithm [150], using a chemical accuracy of 1.6 mHa. The shown space–time requirements are based on the architecture designs considered in our work. Here, we assume that each Toffoli gate is converted to four $T$ gates, meaning that the $3.2 \times 10^{10}$ Toffoli count reported in Ref. [150] is equivalent to executing $1.3 \times 10^{11}$ $T$ gates for a quantum circuit involving 696 logical qubits. Each point in the plots represents a Pareto front solution for the space–time costs considering different MSF sizes, with the leftmost point representing the case where the MSF is large enough to allow serial execution without any idling [62]. The shaded area represents the region for the expected estimates for quantum hardware with specifications between the target and desired ones. We note that Lee et al. [150] report estimates comprising four million physical qubits and a runtime of four days assuming an architecture design based on CCZ factories [175] and hardware parameter values pertaining to $\Lambda = 6.1$ (or $\Lambda = 5.0$) to implement the same logic circuit. The physical runtime of implementing the circuit is determined primarily by how fast magic states can be consumed in executing the non-Clifford gates. A control system reaction time of 10μs was assumed for consuming a CCZ magic state in Ref. [150], and ~10μs is also required for consuming a $T$ magic state in the architecture considered in our work.

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 famil-

iar 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 2 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 heterogenous 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 [177], Qiskit [178], Cirq [179], Pennylane [180], and Classiq [181]) 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

Figure 32: 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.

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[177] (see Appendix G for details). Pennylane Lightning, as another example, provides state vector simulators that can be executed on both AMD and Nvidia GPUs[182].

To enable seamless invocation of quantum kernels from different quantum SDKs within HPC applications developed in C/C++/Fortran, an 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 32 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 and CUDA-Q) 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 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 VI A, 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 VI A 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 massively parallel HPC applications which could allow large-scale system modeling and computation with large datasets; 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) [183, 184].

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 [181]. One example consists of solving linear systems of equations with the Harrow–Hassidim–Lloyd (HHL) algorithm [185]. 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 [186]. 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 33 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 [187], Quil [188], 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 [189, 190].

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) [191] 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,

Figure 33: 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.

can be executed on target quantum devices or simulator backends in a quantum-backend-retargetable manner (see Appendix G for details).

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 communication between distributed 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 next section.

### B. A programming framework for heterogeneous quantum–classical supercomputing

A major challenge in heterogeneous quantum—classical computing is the development of software platforms that enable efficient programming of quantum–HPC systems. Many near- and mid-term algorithms and workflows require the simultaneous and efficient utilization of CPUs, GPUs, and QPUs. Achieving this level of performance requires accurate resource estimation and the development of schedulers capable of balancing algorithms and workloads across accelerators to maximize overall system utilization. One major obstacle is the access to hardware specifications and software libraries developed by quantum hardware vendors, which complicates integration. Additionally, the lack of standardized interfaces and protocols further hinders the seamless integration of QPUs into existing HPC systems.

Although various quantum–HPC platforms could be developed, those designed for near-term critical applications (e.g., quantum error correction) must be built to inherently support low-latency execution and tight coordination between GPUs and QPUs. A key concept for such software platforms is the use of kernels as units of work that can be executed on either GPUs or QPUs and efficiently transferred between them. CUDA-Q is a particularly promising software solution due to its architecture, which natively supports GPU-based acceleration and kernel execution. This makes it well-suited for a tightly coupled heterogeneous quantum–GPU system, like DGX Quantum, that requires efficient resource allocation and load balancing across quantum and classical resources.

The ability to perform mid-circuit measurements is an

Figure 34: 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.

important component of a programming framework for hybrid quantum–classical computing, and has various use cases. Dynamic circuit programming is one such use case, where real-time classical feedback is needed to execute a quantum circuit based on mid-circuit ancillary qubit measurements. Another important example is error mitigation algorithms that rely on mid-circuit measurements, such as quantum subspace expansion (QSE) [192] and zero-noise extrapolation (ZNE) [193]. In QSE, mid-circuit measurements project the quantum state onto a set of basis states, improving the estimation of observables by expanding the solution space. ZNE, on the other hand, uses mid-circuit measurements to monitor error rates and extrapolate results to the zero-noise limit by dynamically adjusting circuit parameters. Another set of use cases are adaptive circuit knitting techniques, which are discussed in the following section. CUDA-Q enables dynamic circuit programming and mid-circuit measurements, and moreover can efficiently program DGX Quantum, which excels in the low-latency communication between QPUs and HPC that are required for these use cases.

### C. 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 small with respect to the number of logical 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 [61]. 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 [194–196]. 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, 197]. 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 [198–200] and even demonstrated a real-time classical interconnect between two superconducting QPUs [201], 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 34. ACK can

Figure 35: Space-time Heterogeneous Circuit Knitting (SHCK) framework. This diagram shows the main steps in the workflow: initial sub-circuit partitioning, finding optimal cut sites in space and time for heterogeneous execution, performing wire cutting, and assigning sub-circuits to the execution of QPUs, GPUs, and CPUs. The last step is observable reconstruction using circuit knitting, which is accelerated by execution on HPC.

be understood as layer-wise circuit learning [202, 203] 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 [204], over a given initial partitioning choice which can be adaptively optimized via feedforward mechanisms.

In Section VI B, 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 VI A. 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 VI B each instance of a disordered quantum many-body system is partitioned separately. 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 2, 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.

As a generalization of ACK, we introduce Space-time Heterogeneous Circuit Knitting (SHCK), shown in Figure 35, as an approach to dynamically cut and merge circuits in both space and time and allocate sub-circuits executions across various quantum and classical accelerators. Key differences in this approach include the use of AI to identify optimal cut sites and the use of quantum wire cutting rather than gate cutting. Wire cutting is a more powerful technique than gate cutting, as it allows for the cutting of a quantum circuit across both time and space. Gate cutting, in contrast, is restricted to decomposing only two-qubit gate operations. The execution of sub-circuits is performed in parallel on QPUs, GPUs, and CPUs. The target for execution of a sub-circuit (i.e. QPU or the type of circuit simulator) depends on the underlying property of a sub-circuit. In this approach, highly entangled sub-circuits are assigned to execution on a QPU and less entangled sub-circuits on a GPU or CPU, e.g. using CUDA-Q simulators. One way to

estimate whether a sub-circuit can be simulated classically is to perform a symbolic simulation of the tensor network without actual execution in order to estimate a bond dimension. Finding optimal cut sites could be achieved using von Neumann quantum entropy calculations combined with AI. Effective load balancing between QPU, GPU, and CPU tasks is a critical component of this workflow to ensure efficient utilization of both quantum and classical resources and to minimize overall execution time.

## D. 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 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.



Figure 36: 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 37: 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.

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 36.

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 37 represents an example scenario in which one or more classical applications running on multiple nodes share the qubits in all/none fashion. This idea 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 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 [205]. In a simple scenario, two heterogeneous jobs submitted to an execution queue each require classical and quantum computing resources. Once resources are available, both the classical

and quantum parts of a 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 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.

### E. Algorithm design for hybrid quantum–classical computing

The design of quantum computers and the impact of quantum computing on utility-scale applications remain poorly understood with much room for improvement. Despite great strides, quantum processors remain small and fragile. There are now dozens of examples of quantum or hybrid quantum–classical algorithms with a proven or expected quantum advantage, but definitive analysis of their performance in practice is elusive, in part because we lack the quantum hardware necessary and simulation suffers from exponential overhead. As hardware matures, we expect that quantum and hybrid algorithms will have greater traction in applications.

Utility-scale quantum computers will inevitably contain numerous classical components for control, calibration, decoding and Pauli frame updates, and support for quantum information transfer through quantum teleportation. At utility scale, many, if not most, algorithms utilizing quantum computing will also use classical computing in a prominent way, not only supporting quantum computation, but also taking a primary role in the computing itself. Hybrid algorithms will dominate well into the era of FTQC, and in many cases will remain the best-performing algorithms in the long term. For some computational tasks such as sorting [78], it has been proven that there can be no quantum advantage even for error-free computation. For many quantum algorithms whose runtime is insensitive to inputs, we do not expect quantum advantage in many conceivable applications, e.g., for quantum search [206], because existing classical techniques tend to show better-than-worst case performance in important cases. Such tasks will continue to be addressed by classical computing.

Even in the case of a proven asymptotic quantum advantage, this advantage might not survive at the utility scale due to large overhead. For example, it has been shown that proven quadratic speedups of many quantum algorithms based on Grover's amplitude amplification (for computational tasks such as, e.g., search, optimization, Monte Carlo methods, and various machine learning tasks), will not result in a quantum advantage on early generations of FTQC devices due to the massive overheads associated with QEC (see [207]). Furthermore, classical logic gates will likely remain orders of magnitude faster and smaller than logical quantum gates even after significant advances in hardware and QEC. Quantum resources, including logical qubits and gates, are also expected to remain costly and limited well into the FTQC era. All of these factors have implications for algorithm design.

While programs that run quantum algorithms usually comprise multiple gates being executed simultaneously, research on distributed quantum computing hardware is still in its infancy and will be a critical area of research going forward. Some design criteria include advancing classical and quantum algorithms jointly; taking advantage of decades' worth of knowledge in classical algorithms and HPC; considering constants and logarithmic factors that are often ignored due to the use of asymptotic big O complexities; identifying common quantum subroutines for efficiency improvements; carefully assessing the costs of accessing and incorporating classical data and information into quantum subroutines and vice versa; including FTQC considerations when analyzing quantum algorithmic impact; exploring the simplest approaches that might work, with an eye to ruling them in or out; and engaging in the co-design of algorithms, error correction, and hardware, both quantum and classical. We use some of these co-design criteria for electronic-structure quantum simulations on FTQC in Section IV and for distributed quantum simulations of strongly disordered systems in condensed phase in Section VI. Here, we briefly discuss additional examples in areas beyond simulation of quantum systems.

**Beyond quadratic speedups in combinatorial optimization.** In principle, subroutines of many classical search algorithms can be replaced by quantum subroutines with a provable quadratic quantum speedup. An example of such a case is when the overall classical algorithm takes advantage of structure but there is an absence of any known structure, or an ignorance as to how to take further advantage of structure, at the subroutine level. However, one must be careful to ensure access to the classical data is well accounted for. As one example, one of the authors of the present paper has

analyzed providing various levels of quantum assistance to state-of-the-art approaches to constraint programming [208]. There is concern that because of large constant factor overheads, a quadratic speedup will not be useful except possibly in the far fault-tolerant regime [207]. Quartic speedups, however, do not suffer the same problem. There is recent analytical work that indicates quartic or even exponential speedups may be possible for some combinatorial optimization problems [209, 210], among other numerical observations of possible favorable scaling for hybrid quantum–classical approaches to combinatorial optimization [211]. There is promising further work in this direction: extending the range of techniques providing such speedups, understanding how to incorporate such techniques in hybrid quantum–classical approaches, and determining how best to do so in a distributed fashion. Many groups continue exploring QAOA and other variational algorithms, examples of the simplest algorithms that might work. Some of these techniques have now been shown to be simulable classically and thus ruled out, while hope remains for certain variants. Recent work on iterative quantum algorithms [212, 213] takes advantage of more-sophisticated classical approaches including noise-directed adaptive remapping, mid-circuit measurements, and inspiration from quantum control theory [214–218], with ties to analog quantum algorithms [219].

**Common subroutines.** Many algorithms that take advantage of quantum computation require quantum versions of classical routines. Such building blocks include the implementation of elementary numerical functions that have well-controlled numerical error, are uniformly scalable and reversible, and can be implemented efficiently. Prior work has provided explicit quantum circuits for some such subroutines for a variety of numerical functions important in scientific computing [220, 221]. Developing explicit circuits for other subroutines, creating distributed versions of these subroutines for hybrid architectures, and understanding when to employ classical vs. quantum computing are key directions for realizing the impact of quantum computing. As another example, quantum Fourier transforms over a variety of groups arise in many algorithms. Further advances in distributing quantum Fourier transforms over realistic architectures and generally improving their efficiency [222] are promising directions of study.

**Quantum-assisted and quantum-ready machine learning.** Machine learning and AI are rapidly advancing fields. A hybrid approach can contribute to advancing these fields for use today while providing insight into the potential impact of quantum computing on these approaches in the longer term. This is an area for "quantum-ready" algorithms, with purely classical algorithms that can be run today for which there are identified subroutines that can be substituted by quantum subroutines. Examples of this approach include quantum-assisted variational autoencoders (QVAE) [223, 224], quantum-assisted associative adversarial networks (QAAAN) [225], combin-

ing energy-based models (EBM) [226] with invertible flows (IF) [227], and quantum-compatible discrete deep generative models [228]. A quantum-ready approach is widely applicable, from optimization [229, 230] (see Section VII) to simulating quantum systems (see Section VI). These quantum-ready approaches are natural targets for present-day and near-term quantum-inspired hardware acceleration.

### F. Toward performance benchmarking of hybrid quantum–classical computers

Here, we discuss how one can approach a holistic benchmarking of hybrid quantum–classical computers, as opposed to the functional benchmarking of critical components of such computers discussed earlier in this paper. Ideally, all performance benchmarking would be hardware-agnostic. This may be satisfied for utility-scale fault-tolerant computers, but, as we will discuss, for mid-scale performance benchmarking some departure from hardware agnosticism may be necessary. We can learn from the field of HPC, where various performance benchmarks aimed at assessing performance relevant to different sorts of computation have been developed. Examples include: dense linear algebra (HPL (High-Performance Linpack) [231, 232]) that serve as the basis for the TOP500 list [232, 233] as well as the Green500 list [207, 232] which ranks supercomputers based on energy efficiency; sparse linear algebra (HPCG (High-Performance Conjugate Gradient) [234, 235]); graph and large scale data analysis (Graph500 [236]); artificial intelligence and machine learning (MLPerf [237]); physical modeling (SPEC HPC Benchmarks [238]); and scientific workloads (NPB (NAS Parallel Benchmarks [239, 240]).

We can consider how to leverage these benchmarks in expanding performance benchmarking of computations that likely require quantum computation, while keeping the benchmarks themselves hardware agnostic, in line with standard HPC benchmarks. Some proposals exist for algorithmic-level benchmarking [241], but much work remains to be done in that area, as well as for mid-scale benchmarking. Even though it will be some time before we can use utility-scale benchmarks for supercomputer-scale hybrid quantum-classical computers, the design of such benchmarks can inform the design of mid-scale benchmarks and provide targets for resource estimation. Such work can be viewed as a natural extension of identifying utility-scale problems with high impact, such as those in Refs. [242–251] from the DARPA QB program [252].

In the meantime, as quantum hardware matures, it is useful to have families of benchmarks that can both guide and gauge hardware and algorithm development. Ideally, these benchmarks would have tunable size and difficulty, enabling them to serve as stepping stones to performance benchmarking of fully fault-tolerant quantum hardware. Some examples for such tunable sets of bench-

mark instances exist [253–255], including those developed in the context of benchmarking quantum-inspired hardware as part of the DARPA Quantum-Inspired Classical Computing (QuICC) program. Such examples can serve as models for the design of families that cover other aspects of computation that will be needed for future applications of quantum computing. As for benchmark families derived from subroutines that we expect to use in hybrid quantum–classical computing (e.g., quantum Fourier transforms, scientific computing subroutines, sampling, search, and error correction), they cannot be completely hardware agnostic in their design, but nevertheless can serve as benchmarks for a diversity of hardware.

It is critical that these benchmarks can assess the performance of the entire computation, including the both quantum and classical components used in solving a problem instance. Moving the field from an exclusive focus on quantum processing time to also considering classical processing time has been recently addressed in Ref. [256], but more work needs to be done in this direction. We can learn from HPC's need to design benchmarks and metrics appropriate for heterogeneous supercomputers of the future; such efforts can inspire benchmarks and metrics for hybrid quantum–classical computers. For both traditional supercomputers and hybrid quantum–classical supercomputers, we expect to see increasing evaluation against multiple metrics such as energy, ratios of computational power to SWaP (size, weight, and power), and amount of computation per time interval. These benchmark sets can not only provide targets for hardware and algorithm development along with assessment of these developments, but can also help identify bottlenecks and set priorities for hardware and algorithm development.

## 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 quantum workload distribution algorithms 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 [177] 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. 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

$$H = -J \sum_{\langle i,j \rangle} \sigma_i^z \sigma_j^z - g \sum_{i=1}^{N} \sigma_i^x, \tag{11}$$

where $\langle i,j \rangle$ denotes all nearest-neighbor pairs in the lattice, $\sigma^z$ and $\sigma^x$ 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 [257].

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 [258, 259]. 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

$$|\psi(t)\rangle \approx \left( \prod_j e^{-ia_j H_j t/r} \right)^r |\psi(0)\rangle, \tag{12}$$

where $t/r$ is the time step 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, for example,

$e^{-i(A+B)t} \approx (e^{-iAt/r}e^{-iBt/r})^r$ 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 38 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

$$\lambda(t) = -\lim_{N\to\infty}\frac{1}{N}\log\mathcal{L}(t), \quad (13)$$

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

While there have been recent demonstrations simulating DQPTs on both quantum devices (a subset of qubits in a 22-qubit superconducting chip [260] and 53 qubits in a trapped ion experiment [261]) as well as using numerical classical simulators [257], 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.



Figure 38: 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 time steps executed on 512 A100 GPUs across 128 nodes on Perlmutter.

We performed several simulations of 2D spin lattices on various compute configurations ranging from a single CPU to 512 GPUs across 128 nodes. Figure 38 shows a DQPT discovered during the simulation of the largest system studied, an 8×5 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. Although we did not perform one, an MPS time-evolution simulation could provide a useful comparison for classical computing resources.



Figure 39: Performance of CUDA-Q simulation using multi-threaded CPU, single GPU, and multi-GPU backends. Systems simulated are 2D lattices: 5×4, 5×5, 5×6, 5×7, and 5×8 qubits. Simulation time reported is for one time step in the time-evolution circuit (100 total time steps).

Circuit synthesis time for one Trotter time step ranges from 0.2–0.4 ms for 5–10 qubits to 2–5 ms 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 39 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 0.04–4 s for 20–25 qubits on a single A100 GPU to 23–33 s 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 600× 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 VI B), 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 V C 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 [194, 195]. 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. [262] 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 40: Schematic of an adaptive circuit knitting method using tensor networks. In the inner loop, a variational optimizer finds in parallel circuit parameters for partitions of a quantum system that are induced by an initial TN ansatz. In the outer loop, an adaptive procedure finds cuts which minimize entanglement between partitions. After the best cuts are found, quantum observables are reconstructed via circuit knitting. The procedure is repeated until a desired accuracy of some (global) observables is achieved or the performance of circuit knitting is plateaued.

Figure 40 shows 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. [262] 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, for example, the dynamical quantum phase transitions discussed in Section VI A. As a prototype, 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

$$H = -\sum_{i=1}^{N-1} J_{i,i+1}\sigma_i^z\sigma_{i+1}^z - \sum_{i=1}^{N} g_i\sigma_i^x - \sum_{i=1}^{N} h_i\sigma_i^z \quad (14)$$

where $\sigma^z$ and $\sigma^x$ 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 hope to exploit for more efficient simulation.

Figure 41 provides a summary of the results for an ensemble of 32-qubit spin chains, each time-evolving under a Hamiltonian with different sets of parameters $J_{i,i+1}$, $g_i$, $h_i$ chosen randomly from a uniform distribution on $[-1, 1]$ (excluding 0). For each spin chain, circuit optimization was carried out at eight time steps 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 41(a) gives a schematic for a single instance on a smaller 20-qubit system. Figure 41(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×, while the 75th and 95th percentiles were 59× and 450×, respectively. While not shown in this figure, the gap between adaptive and load-balanced widens during the later time steps, indicating that for longer simulations the benefits of adaptive circuit knitting will increase. As in Section VI A, the `cuStateVec` simulator was employed to enable performant execution of the 32-qubit simulations.



Figure 41: (a) Example of a 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 V C 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 QUANTUM-INSPIRED AND QUANTUM-ASSISTED PROBABILISTIC COMPUTING

The power of randomness in computational algorithms has long been recognized by computer scientists. In particular, Monte Carlo algorithms, first popularized by Ulam and von Neumann and later developed by Metropolis and Teller, have been widely recognized as one of the top ten algorithms of the 20th century. Today, a considerable amount of workloads in combinatorial optimization, computational science, and AI are intrinsically probabilistic. While such workloads could in principle be accelerated by QPUs (e.g., quantum annealing for combinatorial optimization), shorter-term deployment of industry-scale probabilistic computers could be an efficient way to develop software stacks, know-how, and product market fit that would be quantum-ready and thus can be seamlessly integrated by future quantum technologies. Moreover, it is not clear why we should not capitalize on the probabilistic nature of classical many-body systems to significantly enhance our computational models and frameworks. This allows us to develop novel architectures leveraging intrinsically noisy physical systems, thus avoiding emulation of such probabilistic behavior on deterministic Turing machines, which can be expensive. For example, constructing an individual high-quality tunable random number generator could require up to 15,000 transistors *per node* [263]. More importantly, emulating many-body emergent probabilistic phenomena could be exponentially hard with the sequential Markov Chain Monte Carlo techniques, due to their long mixing times to arrive at non-trivial steady states.

Indeed, Feynman, who pioneered the notion of a controllable quantum computer for simulating other quantum systems (in his 1981 talk "Simulating Physics with Computers" [264]), also imagined a probabilistic computer with the same idea: "The other way to simulate a probabilistic Nature ... is by a computer $C$, which itself is probabilistic". Motivated by this observation, a growing community has recently been developing a full-stack probabilistic computing approach [265]. Such probabilistic computers can be envisioned to take a great deal of inspiration from physics, where nature employs microscopic stochastic local dynamics, quasi-sparse graph connectivity, strong heterogeneity, non-equilibrium dynamics, and asynchronous updates (without global clocks) in probabilistic networks, see Fig 42. These networks evolve in a massively parallel manner, leading to emergent many-body phenomena at mesoscopic scales that are computationally rich and hard to simulate with conventional computing paradigms. Similar to GPUs that are designed to

Figure 42: **Physical analogy** between asynchronously interacting (a) bodies and (b) probabilistic bits: both systems are asynchronous, locally interacting with sparse connectivity and massively parallel.



Figure 43: 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.

accelerate ML problems and QPUs specialized to simulate and model many-body quantum systems, many-body interacting probabilistic bits (p-bits) could be engineered to act as native processors for a wide range of probabilistic applications including Monte Carlo algorithms, Bayesian learning and inference [266], training and inference in energy-based models [226, 267], combinatorial optimization, modeling complex systems, efficient linear algebra [268], accelerating certain quantum simulations, and complex reasoning in generative AI [269].

While p-bits cannot substitute for qubits due to the well-known "sign problem" [270], the boundary between probabilistic and quantum approaches is often blurry and continuously evolving. Specialized p-bit hardware and advanced sampling techniques can help push the boundaries of classical simulability, making these efforts highly worthwhile. 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 [263, 271–274]. 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 high-performance heterogeneous hybrid quantum–classical hardware architectures, as we discuss in Section VII D.

### 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:

$$m_i = \text{sgn}(\tanh(\beta I_i) - r_U), \qquad (15)$$

$$I_i = \sum_j J_{ij} m_j + h_i, \qquad (16)$$

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

There are two immediate generalizations that can be made. One is that p-bits can be extended to have multiple states. These Potts spins have also been implemented in hardware [275] and shown to have better embedding

than simple p-bits for certain optimization problems such as graph coloring. The other is that 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)

$$I_i = \sum_j J_{ij} m_j + \sum_{j<k} J_{ijk} m_j m_k + \sum_{j<k<l} J_{ijkl} m_j m_k m_l, \qquad (17)$$

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. [274] demonstrates an implementation with $k = 3$ 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 [276, 277]. In the proposed approach [276, 277], 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 43. 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 43 with $N = 4$, $y_i = (x_1 \lor x_2 \lor \neg x_4)$, 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 [278], and reduced hardware resource utilization by $O(k^2)$.

### 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 [263, 279], 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 [280]. 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 [263, 271–274, 281]. Even single FPGA-based implementations of p-computers have shown competitive performance against the state of the art [274].

### C. Scaling up p-computers: A distributed approach

Single processing elements (PE), such as FPGAs/GPUs/TPUs, are often not large enough to encode practical problem instances with 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 44). Preliminary results show that as long as the communication links are faster than individual p-bit clocks, distributed probabilistic computers can create the "illusion" [282] of a single PE that can house a much larger graph. Sampling rates over 1000 flips per *nanosecond* are feasible (manuscript in preparation), which bodes well for hard combinatorial optimization and sampling problems.

More generally, many workloads in probabilistic AI models, such as modern energy-based models (EBM) [273, 283], involve several linear algebra operations, mainly as matrix multiplies. Thus, to scale up



Figure 44: 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.

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 45 illustrates a heterogenous system architecture with multiple GPUs and FPGAs that can be used for scaling up to a large number of p-bits (e.g., >1B) enabling it 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 communicates with the FPGA it should access the main memory through PCIe, as shown in the orange flow of Figure 45 leading to significant overhead due to back-and-forth communications. Architectures with Peer2Peer (P2P) communication [284] and disaggregated memory [285, 286] could potentially limit such bottleneck as shown in the green logical flow of Figure 45, 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 [287].

Figure 45: 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.

### 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 44), 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 [288, 289], or digital, based on Quantum Approximation Optimization Algorithm (QAOA) [82, 83]. 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 [230] or quantum-assisted genetic algorithms [290].

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 [80]. 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 [229], 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 transition). 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 [291].

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 45. In some implementations, both quantum and classical processors can be placed on the same chip to get additional performance benefits [292] (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 [293, 294] 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.  TOWARD A COST-EFFECTIVE QUANTUM COMPUTING TECHNOLOGY

A utility-scale quantum computer should be characterized as a machine whose computational value surpasses its total cost [6]. 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. It is widely expected that initial quantum computers will be exorbitantly expensive, far beyond utility scale. Therefore we must analyze the rate at which the costs decrease for a quantum computer to cover more utility scale problems. We first describe the systems engineering considerations for reducing the cost of a quantum supercomputer. Then we discuss how the supply chain for the quantum supercomputer will reduce the costs of R&D and manufacturing

## A. Systems engineering

Based on the resource estimates from FeMoco, as a widely studied candidate for beyond-classical quantum simulation (see Section IV, high-level estimates suggest that a utility-scale quantum computer would require 10-50 quantum accelerators with each accelerator requiring between 100 thousand to 1 million qubits. These are rough estimates, and there is a fair amount of uncertainty due to the qubit quality, algorithmic improvements, technology that will be used to network the quantum accelerators, etc. An optimistic estimated cost to build each accelerator would be on the order of 10 million USD, hence one can imagine that a quantum supercomputer will cost on the order of 100 million to 1 billion USD.

Today, the costs of superconducting qubits are dominated by coaxial wiring, dilution refrigerators (due to size and power of circulators), and quality of the qubits. For example, a superconducting qubit quantum computer with 150 qubits roughly requires 4 million USD in just wiring, and 1 million USD for the dilution refrigerator. Brute force scaling to a 100k qubit system would be cost prohibitive. One key aspect of our approach is that the coaxial wiring can be eliminated through a combination of flex-wiring and the wafer scale wiring wafer. The quality of the qubits is also a key factor because the number of logical qubits that can be yielded is primarily driven by the quality of the qubits.

Additional aspects of our system can drastically reduce costs: cryogenic control leveraging off the shelf CMOS processes can reduce the cost per RF channel. Also, by leveraging existing HPC infrastructures, one can offload costly classical computation such as quantum error correction. For example, Table VIII shows that the computational power of the Nvidia DGX Quantum grows significantly in one generation.

## B. Supply chain management

The cost of a quantum computer encompasses not only manufacturing and operational expenses but also the amortized research and development investments. 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 reduce 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 reduce the cost of the various components needed to build a quantum supercomputer. Much of the reduction of costs of semiconductor products is a result of intense competition, therefore quantum computers that leverages the competitive semiconductor supply chain we will also see a reduction in costs over time. 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 can be compared with operating GPU racks in terms of sophisticated cooling requirements and energy-intensive operation. For example, today's superconducting quantum computers need roughly 20 kW to operate, most of which is consumed by the dilution refrigerator and microwave electronics. Therefore, it is also imperative to build supply chains to lower the operating burden of quantum computers.

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.

## IX. CONCLUDING REMARKS

The ultimate goal of developing a utility-scale quantum computer is to deliver valuable computations that justify the device's cost. Achieving this objective for any architecture involves demonstrating several non-trivial criteria: (1) the existence of a set of computations that produce measurable value when executed; (2) the cost of a

quantum computer capable of performing these computations is lower than the value they generate; and (3) no alternative methods – whether classical, analog, quantum, experimental, etc – can execute these computations more cost-effectively. Evaluating any proposal for a utility-scale quantum computer, including the vision we present here, is thus a challenging task involving answering questions spanning domain-specific use cases, device architecture and integration, and the full range of computational methods. This work focuses primarily on the second criterion, providing an initial holistic understanding of the size, scale, and cost of a quantum supercomputer and identifying key challenges.

We have outlined the design of a quantum supercomputer that combines the latest advances in HPC and superconducting qubits. A comprehensive list of known technical challenges has been used to illuminate obstacles for scaling quantum processors that have been overlooked or addressed piecemeal in prior research [7–9, 90]. In particular, contemporary practical quantum computing investigations have primarily concentrated on technical hurdles at the hundred-qubit level, constrained by the quality of individual qubits. By anticipating technical scaling challenges from hundreds to millions of qubits, our study outlines a holistic system that could provide practical quantum advantage.

For qubit hardware, our thesis is that 300-mm semiconductor processing can radically improve qubit quality, reliability, and scaling. Reproducibility and yield can be greatly improved by replacing junction fabrication from lift-off to a CMOS standard deposition and etch process. Our architecture uses wafer-scale integration, where 20k-qubit modules are connected to 3 Kelvin control circuitry through a 300-mm wiring die. Additionally, an on-chip measurement system eliminates the need for expensive and large volume circulators and pre-amplifiers. Tiling multiple 20k-qubit modules allows fitting 160k qubits in a single DR, and reduces the need for optical interconnects between DRs. This integrated and modular approach thus enables a lower-cost path to scaling.

We introduced a design for a classical supercomputing system that integrates fault-tolerant quantum computing to realize hybrid quantum-classical workloads. HPC is changing, becoming more heterogeneous in response to the need to integrate specialized classical processors. Incorporating quantum processors is just a further step in this trend. We described a hybrid quantum-classical full stack that can be extended as these technologies evolve. The stack is modularized within six layers to allow different software vendors to innovate in parallel. We discussed extensions to existing HPC programming environments and explored various circuit knitting techniques for distributing quantum workloads across multiple QPUs as an alternative to direct quantum networking. We described approaches to designing algorithms that leverage both quantum and classical computing capabilities in close coordination, and discussed extensions to current HPC performance benchmarking standards. At the

lower layers of the stack, we detailed FTQC protocols and tight integration of the HPC system with the quantum control digital logic via low-latency and high-bandwidth PCIe-based interfaces. This allows seamless, standardized, and high-performance integration between QPUs, CPUs, GPUs, and any other compute resource in a heterogeneous HPC system. By minimizing communication overheads, our design for a high-performance quantum–classical stack enables calibration, quantum error correction and characterization, adaptive circuit knitting, and any other hybrid process to be performed at the cutting-edge limit of fidelity and runtime.

We also assess our design with detailed resource estimates of classically-hard electronic structure calculations leveraging detailed experimentally-motivated simulation and benchmarking of fault-tolerant quantum computing protocols using superconducting qubits. We showed that for quantum simulations of FeMoco with chemical accuracy $\epsilon = 1.6$ mHa, one requires hundreds of millions of physical qubits and a minimum of 1.4 years to run with current hardware quality (the 'baseline' hardware in our study). The runtime can be reduced to approximately four months and the qubit count to tens of millions if the hardware quality is significantly improved to our proposed targets. Our sensitivity analysis revealed that improving the gate fidelities would improve the exponential error suppression of surface codes significantly more than reducing SPAM errors or protecting idling qubits. Another enlightening result of our study is that the logical fidelity of surface code depends considerably on the distribution of qubit errors within the qubit array. Moreover, we demonstrate a threshold behavior at $\sim 6\%$ for rows of weaker stabilizer measurements representing capacitive tile-to-tile interconnects, or any anticipated future QPU-to-QPU quantum networking technology (such as optical interconnects). Nevertheless, achieving this fidelity especially at the high Bell-pair generation frequencies required for lattice surgeries between high-distance codes is ambitious for the existing optical interconnect technologies for superconducting qubits. We therefore recommend alternative techniques such as partial fault-tolerant compilation and circuit knitting to be explored in parallel.

Our quantum supercomputing vision ultimately tightly integrates quantum compute, classical compute, and control components at scale using state-of-the-art, yet existing underlying technologies. Not all of the challenges we laid out have been addressed by our approach. Our hope is that, by publicizing the obstacles to scaling quantum computers along with a baseline design, 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.

[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. Gowrishankar, 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. Andersen, M. Ansmann, F. Arute, K. Arya, A. Asfaw, N. Astrakhantsev, J. Atalaya, et al., Nature (2024), 10.1038/s41586-024-08449-y.
[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] M. B. Taylor, IEEE Micro **33**, 8 (2013).
[38] N. P. Breuckmann and J. N. Eberhardt, PRX Quantum **2** (2021), 10.1103/prxquantum.2.040101.
[39] Y. Akahoshi, K. Maruyama, H. Oshima, S. Sato, and K. Fujii, PRX Quantum **5**, 010337 (2024).
[40] R. Toshio, Y. Akahoshi, J. Fujisaki, H. Oshima, S. Sato, and K. Fujii, "Practical quantum advantage on partially fault-tolerant quantum computer," (2024), arXiv:2408.14848 [quant-ph].
[41] D. Gottesman, "Opportunities and challenges in fault-tolerant quantum computation," (2022), 2210.15844.
[42] G. F. Viamontes, I. L. Markov, and J. P. Hayes, in *2007 International Conference on Computer-Aided Design, ICCAD 2007, San Jose, CA, USA, November 5-8, 2007*, edited by G. G. E. Gielen (IEEE Computer Soci-

ety, 2007) pp. 69–74.

[43] K. N. Patel, J. P. Hayes, and I. L. Markov, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **23**, 1220 (2004).

[44] A. Maksymov, J. Nguyen, V. Chaplin, Y. S. Nam, and I. L. Markov, in *IEEE International Symposium on High-Performance Computer Architecture, HPCA 2022, Seoul, South Korea, April 2-6, 2022* (IEEE, 2022) pp. 387–399.

[45] "Testing and debugging quantum programs: The road to 2030," .

[46] 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).

[47] S. Yamashita and I. L. Markov, Quantum Inf. Comput. **10**, 721 (2010).

[48] A. Maksymov, J. Nguyen, V. Chaplin, Y. S. Nam, and I. L. Markov, in *Proc. HPCA* (2020) pp. 387–399.

[49] K. Romanik, Theoretical Computer Science **188**, 79 (1997).

[50] S. A. Metwalli and R. V. Meter, (2023), arXiv:2311.18202 [quant-ph].

[51] M. Caleffi, M. Amoretti, D. Ferrari, J. Illiano, A. Manzalini, and A. S. Cacciapuoti, Computer Networks **254**, 110672 (2024).

[52] B. S. Lee, B. Kim, A. P. Freitas, A. Mohanty, Y. Zhu, G. R. Bhatt, J. Hone, and M. Lipson, Nanophotonics **10**, 99 (2020).

[53] A. Youssefi, I. Shomroni, Y. J. Joshi, N. R. Bernier, A. Lukashchuk, P. Uhrich, L. Qiu, and T. J. Kippenberg, Nature Electronics **4**, 326 (2021).

[54] F. Lecocq, F. Quinlan, K. Cicak, J. Aumentado, S. Diddams, and J. Teufel, Nature **591**, 575 (2021).

[55] D. Awschalom, K. K. Berggren, H. Bernien, S. Bhave, L. D. Carr, P. Davids, S. E. Economou, D. Englund, A. Faraon, M. Fejer, *et al.*, Prx Quantum **2**, 017002 (2021).

[56] R. Delaney, M. Urmey, S. Mittal, B. Brubaker, J. Kindem, P. Burns, C. Regal, and K. Lehnert, Nature **606**, 489 (2022).

[57] S. Sunami, S. Tamiya, R. Inoue, H. Yamasaki, and A. Goban, "Scalable networking of neutral-atom qubits: Nanofiber-based approach for multiprocessor fault-tolerant quantum computer," (2024), arXiv:2407.11111 [quant-ph].

[58] J. Sinclair, J. Ramette, B. Grinkemeyer, D. Bluvstein, M. Lukin, and V. Vuletić, "Fault-tolerant optical interconnects for neutral-atom arrays," (2024), arXiv:2408.08955 [quant-ph].

[59] S. Storz, J. Schär, A. Kulikov, P. Magnard, P. Kurpiers, J. Lütolf, T. Walter, A. Copetudo, K. Reuer, A. Akin, J. Besse, M. Gabureac, G. Norris, A. Rosario, F. Martin, M. J., W. Amaya, M. W. Mitchell, C. Abellan, J. D. Bancal, N. Sangouard, B. Royer, A. Blais, and A. Wallraff, Nature **617**, 265–270 (2023).

[60] E. Giusto, S. Nuñez-Corrales, P. Cao, A. Cilardo, R. K. Iyer, W. Jiang, P. Rech, F. Vella, B. Montrucchio, S. Dasgupta, *et al.*, arXiv preprint arXiv:2408.10484 (2024).

[61] M. E. Beverland, P. Murali, M. Troyer, K. M. Svore, T. Hoefler, V. Kliuchnikov, G. H. Low, M. Soeken, A. Sundaram, and A. Vaschillo, "Assessing requirements to scale to practical quantum advantage," (2022),

arXiv:2211.07629 [quant-ph].

[62] A. Silva, A. Scherer, Z. Webb, A. Khalid, B. Kulchytskyy, 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].

[63] 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).

[64] 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).

[65] V. Giovannetti, S. Lloyd, and L. Maccone, Physical Review Letters **100** (2008), 10.1103/physrevlett.100.160501.

[66] O. D. Matteo, V. Gheorghiu, and M. Mosca, IEEE Transactions on Quantum Engineering **1**, 1–13 (2020).

[67] 1QB Information Technologies (1QBit), "TopQAD: Topological Quantum Architecture Design [Software Documentation]," (2024).

[68] A. W. Harrow, A. Hassidim, and S. Lloyd, Phys. Rev. Lett. **103**, 150502 (2009).

[69] A. M. Childs, R. Kothari, and R. D. Somma, SIAM Journal on Computing **46**, 1920–1950 (2017).

[70] D. Horn and A. Gottlieb, Phys. Rev. Lett. **88**, 018702 (2001).

[71] S. Lloyd, M. Mohseni, and P. Rebentrost, Nature Physics **10**, 631–633 (2014).

[72] P. Rebentrost, M. Mohseni, and S. Lloyd, Physical Review Letters **113** (2014), 10.1103/physrevlett.113.130503.

[73] 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.

[74] 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).

[75] T. Häner, M. Roetteler, and K. M. Svore, "Optimizing quantum circuits for arithmetic," (2018), arXiv:1805.12445 [quant-ph].

[76] T. Häner, M. Soeken, M. Roetteler, and K. M. Svore, "Quantum circuits for floating-point arithmetic," (2018), arXiv:1807.02023 [quant-ph].

[77] R. Babbush, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, A. Paler, A. Fowler, and H. Neven, Phys. Rev. X **8**, 041015 (2018).

[78] 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.

[79] D. P. Rodgers, SIGARCH Comput. Archit. News **13**, 225–231 (1985).

[80] M. Szegedy, in *45th Annual IEEE symposium on foundations of computer science* (IEEE, 2004) pp. 32–41.

[81] R. Babbush, J. R. McClean, M. Newman, C. Gidney, S. Boixo, and H. Neven, PRX Quantum **2**, 010103 (2021).

[82] E. Farhi, J. Goldstone, and S. Gutmann, arXiv preprint arXiv:1411.4028 (2014).

[83] K. Blekos, D. Brand, A. Ceschini, C.-H. Chou, R.-H. Li, K. Pandya, and A. Summer, Physics Reports **1068**, 1 (2024).

[84] 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.

[85] S. Boulebnane and A. Montanaro, PRX Quantum **5**, 030348 (2024).

[86] B. Augustino, M. Cain, E. Farhi, S. Gupta, S. Gutmann, D. Ranard, E. Tang, and K. Van Kirk, arXiv preprint arXiv:2410.03015 (2024).

[87] A. Montanaro and L. Zhou, (2024), arXiv:2411.04979v1, 2411.04979.

[88] D. W. Berry, Y. Tong, T. Khattar, 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].

[89] 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).

[90] K. M. Svore, A. V. Aho, A. W. Cross, I. L. Chuang, and I. L. Markov, Computer **39**, 74 (2006).

[91] C. Liu *et al.*, Phys. Rev. Lett. **132**, 017001 (2024).

[92] 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).

[93] Cerebras, "Wafer scale engine," (2023).

[94] NVidia, "Nvidia blackwell platform arrives to power a new era of computing," (2024).

[95] A. Materials, "Heterogeneous integration," (2024).

[96] A. Opremcak *et al.*, Phys. Rev. X **102**, 011027 (2020).

[97] M. Mirhosseini, A. Sipahigil, M. Kalaee, and O. Painter, Nature **588** (2023).

[98] 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].

[99] M. J. Arshad, C. Bekker, B. Haylock, K. Skrzypczak, D. White, B. Griffiths, J. Gore, G. W. Morley, P. Salter, J. Smith, I. Zohar, A. Finkler, Y. Altmann, E. M. Gauger, and C. Bonato, Phys. Rev. Appl. **21**, 024026 (2024).

[100] D. L. Craig, H. Moon, F. Fedele, D. T. Lennon, B. van Straaten, F. Vigneau, L. C. Camenzind, D. M. Zumbühl, G. A. D. Briggs, M. A. Osborne, *et al.*, Physical Review X **14**, 011001 (2024).

[101] F. Berritta, J. A. Krzywda, J. Benestad, J. van der Heijden, F. Fedele, S. Fallahi, G. C. Gardner, M. J. Manfra, E. van Nieuwenburg, J. Danon, A. Chatterjee, and F. Kuemmeth, Phys. Rev. Appl. **22**, 014033 (2024).

[102] A. Wozniakowski, J. Thompson, M. Gu, and F. Binder, arXiv preprint arXiv:2005.06194 (2020).

[103] J. Schuff, M. J. Carballido, M. Kotzagiannidis, J. C. Calvo, M. Caselli, J. Rawling, D. L. Craig, B. van Straaten, B. Severin, F. Fedele, S. Svab, P. C. Kwon, R. S. Eggli, T. Patlatiuk, N. Korda, D. Zumbühl, and N. Ares, (2024), arXiv:2402.03931 [cond-mat.mes-hall].

[104] B. Severin, D. T. Lennon, L. C. Camenzind, F. Vigneau, F. Fedele, D. Jirovec, A. Ballabio, D. Chrastina, G. Isella, M. de Kruijf, M. J. Carballido, S. Svab, A. V. Kuhlmann, S. Geyer, F. N. M. Froning, H. Moon, M. A. Osborne, D. Sejdinovic, G. Katsaros, D. M. Zumbühl, G. A. D. Briggs, and N. Ares, Scientific Reports **14**, 17281 (2024).

[105] S. S. Kalantre, J. P. Zwolak, S. Ragole, X. Wu, N. M. Zimmerman, M. D. Stewart, and J. M. Taylor, npj Quantum Information **5**, 6 (2019).

[106] J. D. Teske, S. S. Humpohl, R. Otten, P. Bethke, P. Cerfontaine, J. Dedden, A. Ludwig, A. D. Wieck, and H. Bluhm, Applied Physics Letters **114**, 133102 (2019).

[107] V. Sivak, A. Eickbusch, H. Liu, *et al.*, Physical Review X **12**, 011059 (2022).

[108] Y. Kurman, L. Ella, R. Szmuk, O. Wertheim, B. Dorschner, S. Stanwyck, and Y. Cohen, "Benchmarking the ability of a controller to execute quantum error corrected non-clifford circuits," (2024), arXiv:2311.07121 [quant-ph].

[109] F. Berritta, T. Rasmussen, J. A. Krzywda, J. van der Heijden, F. Fedele, S. Fallahi, G. C. Gardner, M. J. Manfra, E. van Nieuwenburg, J. Danon, A. Chatterjee, and F. Kuemmeth, Nature Communications **15**, 1676 (2024).

[110] E. Scerri, E. M. Gauger, and C. Bonato, New Journal of Physics **22**, 035002 (2020).

[111] A. Seif, K. A. Landsman, N. M. Linke, C. Figgatt, C. Monroe, and M. Hafezi, Journal of Physics B: Atomic, Molecular and Optical Physics **51**, 174006 (2018).

[112] L. Phuttitarn, B. Becker, R. Chinnarasu, T. Graham, and M. Saffman, Physical Review Applied **22**, 024011 (2024).

[113] NVIDIA Corporation, "NVIDIA DGX Quantum," https://www.nvidia.com/en-us/data-center/dgx-quantum/ (2025), accessed: 2025-01-22.

[114] NVIDIA Corporation, "NVIDIA CUDA-Q," https://developer.nvidia.com/cuda-q (2025), accessed: 2025-01-22.

[115] P. W. Shor, Physical review A **52**, R2493 (1995).

[116] P. W. Shor, in *Proceedings of 37th conference on foundations of computer science* (IEEE, 1996) pp. 56–65.

[117] J. Preskill, "Fault-tolerant quantum computation," (1997), arXiv:quant-ph/9712048 [quant-ph].

[118] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Journal of Mathematical Physics **43**, 4452–4505 (2002).

[119] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Physical Review A **86** (2012), 10.1103/physreva.86.032324.

[120] H. Bombin and M. A. Martin-Delgado, Phys. Rev. Lett. **97**, 180501 (2006).

[121] A. J. Landahl, J. T. Anderson, and P. R. Rice, "Fault-tolerant quantum computing with color codes," (2011), arXiv:1108.5738 [quant-ph].

[122] B. M. Terhal, Reviews of Modern Physics **87**, 307 (2015).

[123] C. Wang, J. Harrington, and J. Preskill, Annals of

Physics **303**, 31 (2003).

[124] H. Bombin and M. A. Martin-Delgado, Phys. Rev. A **76**, 012305 (2007).

[125] D. Horsman, A. G. Fowler, S. Devitt, and R. V. Meter, New Journal of Physics **14**, 123011 (2012).

[126] B. Eastin and E. Knill, Phys. Rev. Lett. **102**, 110502 (2009).

[127] S. Singh, A. S. Darmawan, B. J. Brown, and S. Puri, Phys. Rev. A **105**, 052410 (2022).

[128] C. Gidney, arXiv preprint arXiv:2302.12292 (2023).

[129] C. Gidney, N. Shutty, and C. Jones, arXiv preprint arXiv:2409.17595 (2024).

[130] S. Bravyi and A. Kitaev, Phys. Rev. A **71**, 022316 (2005).

[131] Z. Chen *et al.*, Nature **595**, 383–387 (2021).

[132] C. Gidney, Quantum **5**, 497 (2021).

[133] O. Higgott, ACM Transactions on Quantum Computing **3**, 1 (2022).

[134] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Physical Review A—Atomic, Molecular, and Optical Physics **86**, 032324 (2012).

[135] I. Quantum, Accessed: 2024-10-09.

[136] T. K. Ho, in *Proceedings of 3rd international conference on document analysis and recognition*, Vol. 1 (IEEE, 1995) pp. 278–282.

[137] C. Chamberland and E. T. Campbell, PRX Quantum **3** (2022), 10.1103/prxquantum.3.010331.

[138] M. A. Nielsen, Physics Letters A **303**, 249–252 (2002).

[139] 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].

[140] C. Gidney, Quantum **8**, 1310 (2024).

[141] Y. Kurman, L. Ella, N. Halay, O. Wertheim, and Y. Cohen, "Controller-decoder system requirements derived by implementing shor's algorithm with surface code," (2024), arXiv:2412.00289 [quant-ph].

[142] P. Panteleev and G. Kalachev, in *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing* (2022) pp. 375–388.

[143] E. Swaroop, T. Jochym-O'Connor, and T. J. Yoder, arXiv preprint arXiv:2410.03628 (2024).

[144] J. Pan, Nature Computational Science **4**, 881 (2024).

[145] Microsoft, "Azure Quantum Resource Estimator," (2022).

[146] A. Szabo and N. S. Ostlund, *Modern quantum chemistry: introduction to advanced electronic structure theory* (Courier Corporation, 1996).

[147] T. D. Crawford, E. Kraka, J. F. Stanton, and D. Cremer, The Journal of Chemical Physics **114**, 10638 (2001).

[148] E. Kraka and D. Cremer, Journal of the American Chemical Society **122**, 8245 (2000).

[149] 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.

[150] J. Lee, D. W. Berry, C. Gidney, W. J. Huggins, J. R. McClean, N. Wiebe, and R. Babbush, PRX Quantum

[151] 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).

[152] A. Y. Kitaev, "Quantum measurements and the abelian stabilizer problem," (1995), arXiv:quant-ph/9511026 [quant-ph].

[153] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2011).

[154] 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.

[155] S. B. Bravyi and A. Y. Kitaev, Annals of Physics **298**, 210 (2002).

[156] 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. Zaribafiyan, (2022), 10.48550/arXiv.2206.12424, arXiv:2206.12424.

[157] 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).

[158] G. H. Low and I. L. Chuang, Quantum **3**, 163 (2019).

[159] D. W. Berry, C. Gidney, M. Motta, J. R. McClean, and R. Babbush, Quantum **3**, 208 (2019).

[160] V. von Burg, G. H. Low, T. Häner, D. S. Steiger, M. Reiher, M. Roetteler, and M. Troyer, Phys. Rev. Res. **3**, 033055 (2021).

[161] 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.

[162] 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).

[163] 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).

[164] T. W. Watts, M. Otten, J. T. Necaise, 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).

[165] 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).

[166] 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).

[167] D. Poulin, A. Kitaev, D. S. Steiger, M. B. Hastings, and M. Troyer, Phys. Rev. Lett. **121**, 010501 (2018).

[168] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan, Rev. Mod. Phys. **92**, 015003 (2020).

[169] gridsynth: An open-source software package to compute approximations of arbitrary-angle z-rotations over the Clifford+T gate set,.

**2**, 030305 (2021).

[170] P. Selinger, Quantum Info. Comput. **15**, 159–180 (2015).

[171] N. J. Ross and P. Selinger, Quantum Info. Comput. **16**, 901–953 (2016).

[172] S. R. White, Phys. Rev. Lett. **69**, 2863 (1992).

[173] H. Gao, S. Imamura, A. Kasagi, and E. Yoshida, Journal of Chemical Theory and Computation **20**, 1185 (2024).

[174] A. Menczer, M. van Damme, A. Rask, L. Huntington, J. Hammond, S. S. Xantheas, M. Ganahl, and Örs Legeza, "Parallel implementation of the density matrix renormalization group method achieving a quarter petaflops performance on a single dgx-h100 gpu node," (2024), arXiv:2407.07411 [physics.chem-ph].

[175] C. Gidney and A. G. Fowler, "Flexible layout of surface code computations using AutoCCZ states," (2019), arXiv:1905.08916 [quant-ph].

[176] D. Rocca, C. L. Cortes, J. F. Gonthier, P. J. Ollitrault, R. M. Parrish, G.-L. Anselmetti, M. Degroote, N. Moll, R. Santagati, and M. Streif, Journal of Chemical Theory and Computation **20**, 4639 (2024), pMID: 38788209, https://doi.org/10.1021/acs.jctc.4c00352.

[177] NVIDIA., "Cuda-q," .

[178] IBM., "Introduction to qiskit," .

[179] Google., "Cirq: An open source framework for programming quantum computers," .

[180] Xanadu., "Pennylane," .

[181] Classiq Technologies Ltd., "Classiq platform," .

[182] 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).

[183] Slurm., "Slurm workload manager," .

[184] OpenPBS., "Open pbs open source project," .

[185] H. Jose Morrell Jr, A. Zaman, and H. Y. Wong, arXiv:2108.09004 (2021).

[186] A. Esposito and T. Danzig, in *2024 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)* (IEEE, 2024) pp. 1088–1094.

[187] A. W. Cross, L. S. Bishop, J. A. Smolin, and J. M. Gambetta, arXiv:1707.03429 (2017).

[188] R. S. Smit, M. J. Curtis, and W. J. Zeng, arXiv:1608.03355v2 (2017).

[189] A. Litteken, Y.-C. Fan, D. Singh, M. Martonosi, and F. T. Chong, Quantum Science and Technology **5**, 034013 (2020).

[190] F. T. Chong, D. Franklin, and M. Martonosi, Nature **549**, 180–187 (2017).

[191] Microsoft., "Quantum intermediate representation (qir) specification," .

[192] N. Yoshioka, H. Hakoshima, Y. Matsuzaki, Y. Tokunaga, Y. Suzuki, and S. Endo, Physical Review Letters **129**, 020502 (2022).

[193] T. Giurgica-Tiron, Y. Hindy, R. LaRose, A. Mari, and W. J. Zeng, in *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)* (IEEE, 2020) pp. 306–316.

[194] S. R. White, Physical review b **48**, 10345 (1993).

[195] H.-D. Meyer, U. Manthe, and L. S. Cederbaum, Chemical Physics Letters **165**, 73 (1990).

[196] U. Manthe, The Journal of chemical physics **128** (2008).

[197] K. Temme, S. Bravyi, and J. M. Gambetta, Phys. Rev. Lett. **119**, 180509 (2017).

[198] 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.

[199] W. Tang and M. Martonosi, arXiv preprint arXiv:2207.00933 (2022).

[200] S. Basu, A. Das, A. Saha, A. Chakrabarti, and S. Sur-Kolay, Journal of Systems and Software **214**, 112085 (2024).

[201] A. Carrera Vazquez, C. Tornow, D. Ristè, S. Woerner, M. Takita, and D. J. Egger, Nature , 1 (2024).

[202] A. Skolik, J. R. McClean, M. Mohseni, P. Van Der Smagt, and M. Leib, Quantum Machine Intelligence **3**, 1 (2021).

[203] 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).

[204] 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].

[205] A. Esposito, J. R. Jones, S. Cabaniols, and D. Brayford, in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, Vol. 2 (IEEE, 2023) pp. 117–121.

[206] G. F. Viamontes, I. L. Markov, and J. P. Hayes, Comput. Sci. Eng. **7**, 62 (2005).

[207] R. Babbush, J. R. McClean, M. Newman, C. Gidney, S. Boixo, and H. Neven, PRX quantum **2**, 010103 (2021).

[208] K. E. C. Booth *et al.*, Quantum **5**, 550 (2021).

[209] S. Chakrabarti, D. Herman, G. Ozgul, S. Zhu, B. Augustino, T. Hao, Z. He, R. Shaydulin, and M. Pistoia, arXiv preprint arXiv:2410.23270 (2024).

[210] S. P. Jordan, N. Shutty, M. Wootters, A. Zalcman, A. Schmidhuber, R. King, S. V. Isakov, and R. Babbush, arXiv preprint arXiv:2408.08292 (2024).

[211] K. Sankar, A. Scherer, S. Kako, S. Reifenstein, N. Ghadermarzy, W. B. Krayenhoff, Y. Inui, E. Ng, T. Onodera, P. Ronagh, *et al.*, npj Quantum Information **10**, 64 (2024).

[212] L. T. Brady and S. Hadfield, arXiv:2309.13110 (2023).

[213] M. Dupont *et al.*, Science Advances **9** (2023).

[214] A. B. Magann, K. M. Rudinger, M. D. Grace, and M. Sarovar, Physical Review A **106**, 062414 (2022).

[215] L. T. Brady and S. Hadfield, arXiv preprint arXiv:2409.15426 (2024).

[216] T. Stollenwerk and S. Hadfield, arXiv preprint arXiv:2403.11514 (2024).

[217] F. B. Maciejewski, B. G. Bach, M. Dupont, P. A. Lott, B. Sundar, D. E. B. Neira, I. Safro, and D. Venturelli, arXiv preprint arXiv:2408.07793 (2024).

[218] F. B. Maciejewski, J. Biamonte, S. Hadfield, and D. Venturelli, arXiv preprint arXiv:2404.01412 (2024).

[219] L. T. Brady, C. L. Baldwin, A. Bapat, Y. Kharkov, and A. V. Gorshkov, Physical Review Letters **126**, 070505 (2021).

[220] S. A. Hadfield, *Quantum algorithms for scientific computing and approximate optimization* (Columbia University, 2018).

[221] S. Hadfield, ACM Transactions on Quantum Computing

**2**, 1 (2021).

[222] E. M. Murairi, M. S. Alam, H. Lamm, S. Hadfield, and E. Gustafson, Physical Review D **110**, 074501 (2024).

[223] N. Gao *et al.*, in *Proc. of KDD '20* (2020).

[224] A. Akbari Asanjan *et al.*, Remote Sensing **15** (2023).

[225] M. Wilson, T. Vandal, T. Hogg, and E. G. Rieffel, Quantum Machine Intelligence **3**, 19 (2021).

[226] P. Huembeli, J. M. Arrazola, N. Killoran, M. Mohseni, and P. Wittek, "The physics of energy-based models," (2022), iSSN: 2524-4906 Issue: 1 Volume: 4.

[227] D. O'Connor *et al.*, arXiv:2012.13196 (2021).

[228] T. Templin *et al.*, arXiv:2303.12302 (2023).

[229] 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).

[230] V. S. Denchev, M. Mohseni, and H. Neven, "Quantum assisted optimization," (2022), uS Patent 11,449,760.

[231] T. Davies, C. Karlsson, H. Liu, C. Ding, and Z. Chen, in *Proceedings of the international conference on Supercomputing* (2011) pp. 162–171.

[232] "Top500: The list of the world's top supercomputers," Accessed: 2025-01-24.

[233] E. Strohmaier, H. W. Meuer, J. Dongarra, and H. D. Simon, Computer **48**, 42 (2015).

[234] "Hpcg benchmark: High performance conjugate gradients benchmark," Accessed: 2025-01-24.

[235] J. Dongarra, M. A. Heroux, and P. Luszczek, National Science Review **3**, 30 (2016).

[236] Graph500, "Graph500," (2024).

[237] NVIDIA, "Mlperf," (2024).

[238] S. P. E. Corporation, "Spec hpc benchmarks," (2024).

[239] "Nas parallel benchmarks (npb)," Accessed: 2025-01-24.

[240] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, *et al.*, The International Journal of Supercomputing Applications **5**, 63 (1991).

[241] T. Proctor, K. Young, A. D. Baczewski, and R. Blume-Kohout, Nature Reviews Physics , 1 (2025).

[242] 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, L. Wang, S. Patel, M. Kodrycka, I. Loaiza, R. A. Lang, A. Aspuru-Guzik, A. F. Izmaylov, J. R. Fontalvo, and Y. Cao, "Feasibility of accelerating homogeneous catalyst discovery with fault-tolerant quantum computers," (2024), arXiv:2406.06335 [quant-ph].

[243] J. Penuel, A. Katabarwa, P. D. Johnson, C. Farquhar, Y. Cao, and M. C. Garrett, "Feasibility of accelerating incompressible computational fluid dynamics simulations with fault-tolerant quantum computers," (2024), arXiv:2406.06323 [quant-ph].

[244] J. E. Elenewski, C. M. Camara, and A. Kalev, "Prospects for nmr spectral prediction on fault-tolerant quantum computers," (2024), arXiv:2406.09340 [quant-ph].

[245] A. A. Agrawal, J. Job, T. L. Wilson, S. N. Saadatmand, M. J. Hodson, J. Y. Mutus, A. Caesura, P. D. Johnson, J. E. Elenewski, K. J. Morrell, and A. F. Kemper, "Quantifying fault tolerant simulation of strongly correlated systems using the fermi-hubbard model," (2024), arXiv:2406.06511 [quant-ph].

[246] E. Mozgunov, J. Marshall, and N. Anand, "Applica-tions and resource estimates for open system simulation on a quantum computer," (2024), arXiv:2406.06281 [quant-ph].

[247] A. Bärtschi, F. Caravelli, C. Coffrin, J. Colina, S. Eidenbenz, A. Jayakumar, S. Lawrence, M. Lee, A. Y. Lokhov, A. Mishra, S. Misra, Z. Morrell, Z. Mughal, D. Neill, A. Piryatinski, A. Scheie, M. Vuffray, and Y. Zhang, "Potential applications of quantum computing at los alamos national laboratory," (2024), arXiv:2406.06625 [quant-ph].

[248] S. N. Saadatmand, T. L. Wilson, M. Field, M. K. Vijayan, T. P. Le, J. Ruh, A. S. Maan, I. Moflic, A. Caesura, A. Paler, M. J. Hodson, S. J. Devitt, and J. Y. Mutus, "Fault-tolerant resource estimation using graph-state compilation on a modular superconducting architecture," (2024), arXiv:2406.06015 [quant-ph].

[249] 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, E. B. Isaacs, S. D. Johnson, L. Mathieson, K. M. Obenland, M. Otten, R. Sundareswara, and A. Holmes, "Quantum computing for corrosion-resistant materials and anti-corrosive coatings design," (2024), arXiv:2406.18759 [quant-ph].

[250] M. Otten, T. W. Watts, S. D. Johnson, R. Sundareswara, Z. Wang, T. S. Hardikar, K. Heitritter, J. Brown, K. Setia, and A. Holmes, "Quantum resources required for binding affinity calculations of amyloid beta," (2024), arXiv:2406.18744 [quant-ph].

[251] T. W. Watts, M. Otten, J. T. Necaise, N. Nguyen, B. Link, K. S. Williams, Y. R. Sanders, S. J. Elman, M. Kieferova, M. J. Bremner, K. J. Morrell, J. E. Elenewski, S. D. Johnson, L. Mathieson, K. M. Obenland, R. Sundareswara, and A. Holmes, "Fullerene-encapsulated cyclic ozone for the next generation of nano-sized propellants via quantum computation," (2024), arXiv:2408.13244 [quant-ph].

[252] DARPA, "Quantum benchmarking," (2024).

[253] S. Mandrà, H. Munoz-Bauza, G. Mossi, and E. G. Rieffel, Future Generation Computer Systems , 107721 (2025).

[254] D. Perera, I. Akpabio, F. Hamze, S. Mandra, N. Rose, M. Aramon, and H. G. Katzgraber, arXiv preprint arXiv:2005.14344 (2020).

[255] E. Rieffel, D. Venturelli, M. Do, I. Hen, and J. Frank, in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 28 (2014).

[256] D. E. B. Neira, R. Brown, P. Sathe, F. Wudarski, M. Pavone, E. G. Rieffel, and D. Venturelli, arXiv preprint arXiv:2402.10255 (2024).

[257] M. Heyl, Reports on Progress in Physics **81**, 054001 (2018).

[258] H. F. Trotter, Proceedings of the American Mathematical Society **10**, 545 (1959).

[259] M. Suzuki, Communications in Mathematical Physics **51**, 183 (1976).

[260] J. Dborin, V. Wimalaweera, F. Barratt, E. Ostby, T. E. O'Brien, and A. G. Green, Nature Communications **13**, 5977 (2022).

[261] 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).

[262] S.-H. Lin, R. Dilip, A. G. Green, A. Smith, and F. Pollmann, PRX Quantum **2**, 010342 (2021).

[263] 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).

[264] R. P. Feynman, International journal of theoretical physics **21**, 467 (1982).

[265] S. Chowdhury, A. Grimaldi, N. A. Aadit, S. Niazi, K. Y. Camsari, *et al.*, IEEE Journal on Exploratory Solid-State Computational Devices and Circuits (2023).

[266] M. Aifer, S. Duffield, K. Donatella, D. Melanson, P. Klett, Z. Belateche, G. Crooks, A. J. Martinez, and P. J. Coles, arXiv preprint arXiv:2410.01793 (2024).

[267] P. J. Coles, C. Szczepanski, D. Melanson, K. Donatella, A. J. Martinez, and F. Sbahi, in *2023 IEEE International Conference on Rebooting Computing (ICRC)* (IEEE, 2023) pp. 1–10, arXiv:2302.06584.

[268] M. Aifer, K. Donatella, M. H. Gordon, S. Duffield, T. Ahle, D. Simpson, G. Crooks, and P. J. Coles, npj Unconventional Computing **1**, 13 (2024).

[269] M. Esencan, T. A. Kumar, A. A. Asanjan, P. A. Lott, M. Mohseni, C. Unlu, D. Venturelli, and A. Ho, "Combinatorial reasoning: Selecting reasons in generative ai pipelines via combinatorial optimization," (2024), arXiv:2407.00071.

[270] S. Chowdhury, K. Y. Camsari, and S. Datta, IEEE Access (2023).

[271] B. Sutton, R. Faria, L. A. Ghantasala, R. Jaiswal, K. Y. Camsari, and S. Datta, IEEE Access **8**, 157238 (2020).

[272] N. A. Aadit, A. Grimaldi, M. Carpentieri, L. Theogarajan, J. M. Martinis, G. Finocchio, and K. Y. Camsari, Nature Electronics **5**, 460 (2022).

[273] S. Niazi, S. Chowdhury, N. A. Aadit, M. Mohseni, Y. Qin, and K. Y. Camsari, Nature Electronics , 1 (2024).

[274] S. Nikhar, S. Kannan, N. A. Aadit, S. Chowdhury, and K. Y. Camsari, Nature Communications **15**, 8977 (2024).

[275] W. Whitehead, Z. Nelson, K. Y. Camsari, and L. Theogarajan, Nature Electronics **6**, 1009 (2023).

[276] 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.

[277] T. Bhattacharya, G. H. Hutchinson, G. Pedretti, X. Sheng, J. Ignowski, T. Van Vaerenbergh, R. Beausoleil, J. P. Strachan, and D. B. Strukov, Nature Communications **15**, 8211 (2024).

[278] D. Dobrynin, A. Renaudineau, M. Hizzani, D. Strukov, M. Mohseni, and J. P. Strachan, Physical Review E **110**, 045308 (2024).

[279] W. A. Borders, K. Y. Camsari, *et al.*, Nature (2019).

[280] 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.

[281] J. Kaiser and S. Datta, Applied Physics Letters **119**, 150503 (2021).

[282] 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).

[283] R. Liao, S. Kornblith, M. Ren, D. J. Fleet, and G. Hinton, arXiv preprint arXiv:2210.10318 (2022).

[284] R. Bittner, E. Ruf, and A. Forin, Cluster Computing **17**, 339 (2014).

[285] K. Lim, J. Chang, T. Mudge, P. Ranganathan, S. K. Reinhardt, and T. F. Wenisch, ACM SIGARCH computer architecture news **37**, 267 (2009).

[286] W. Jiang, M. Zeller, R. Waleffe, T. Hoefler, and G. Alonso, arXiv preprint arXiv:2310.09949 (2023).

[287] 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).

[288] T. Kadowaki and H. Nishimori, Physical Review E **58**, 5355 (1998).

[289] T. Albash and D. A. Lidar, Reviews of Modern Physics **90**, 015002 (2018).

[290] J. King, M. Mohseni, W. Bernoudy, A. Fréchette, H. Sadeghi, S. V. Isakov, H. Neven, and M. H. Amin, (2019), arXiv:1907.00707 [quant-ph].

[291] 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).

[292] M. Mohseni and H. Neven, "Chips including classical and quantum computing processors," (2020), uS Patent 10,671,559.

[293] M. Mohseni, M. M. Rams, S. V. Isakov, D. Eppens, S. Pielawa, J. Strumpfer, S. Boixo, and H. Neven, Phys. Rev. E **108** (2023).

[294] A. Zucca, H. Sadeghi, M. Mohseni, and M. H. Amin, arXiv:2110.10196 (2021).

[295] A. Y. Kitaev, "Quantum measurements and the abelian stabilizer problem," (1995), arXiv:quant-ph/9511026 [quant-ph].

[296] D. S. Abrams and S. Lloyd, Phys. Rev. Lett. **83**, 5162 (1999).

[297] A. M. Childs, Y. Su, M. C. Tran, N. Wiebe, and S. Zhu, Phys. Rev. X **11**, 011020 (2021).

[298] S. Zhuk, N. Robertson, and S. Bravyi, "Trotter error bounds and dynamic multi-product formulas for hamiltonian simulation," (2024), arXiv:2306.12569 [quant-ph].

[299] S. Keller, K. Boguslawski, T. Janowski, M. Reiher, and P. Pulay, The Journal of Chemical Physics **142**, 244104 (2015).

[300] W. Sennane, J.-P. Piquemal, and M. J. Rančić, Phys. Rev. A **107**, 012416 (2023).

[301] D. Poulin, M. B. Hastings, D. Wecker, N. Wiebe, A. C. Doberty, and M. Troyer, Quantum Info. Comput. **15**, 361–384 (2015).

[302] 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.

[303] V. von Burg, G. H. Low, T. Häner, D. S. Steiger, M. Reiher, M. Roetteler, and M. Troyer, Physical Review Research **3**, 033055 (2021).

[304] 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 logic circuits for quantum resource estimation

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 logic circuits, and explain how these bounds must be chosen to ensure that the quantum simulations achieve a given target accuracy. While in our study we use the $p$-benzyne molecule as a concrete example, the described methodology applies to other molecules.

### 1. Workflows for generating logic circuits

The quantum phase estimation (QPE) algorithm [295, 296] 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 $\exp(-iHt)$. More-advanced approaches to electronic-structure quantum simulations are typically based on the framework of qubitization [158]. This framework 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 "qubiterate" 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. [166]). Our quantum resource estimation (QRE) studies report estimates for two algorithmic approaches: the Trotter-based approach, and the qubitization-based double low-rank factorization algorithm. In what follows, we outline the workflow of QRE for the standard QPE algorithm, with the time-evolution operator $\exp(-iHt)$ implemented using Hamiltonian simulation based on the use of product formulae (PF) and Trotterization. The associated analysis of Trotter errors and their propagation into QPE is discussed in Appendix A 2. We refer the reader to the literature for various qubitization-based approaches [61, 150, 159–165, 176]). The propagation of errors in qubitization is briefly outlined in Appendix A 3. A brief description of the double-factorized quantum chemistry simulations used in our QRE studies is given in Appendix C.

In the Trotter-based approaches to QPE, the time-evolution operator $\exp(-iHt)$ is approximated by a PF along with Trotterization. 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 (see Refs. [297, 298])

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

Our resource estimation analyses are based on using either the first-order Lie–Trotter formula or the second-order Trotter–Suzuki formula.

In the framework of second quantization, the electronic model Hamiltonian is typically given as

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

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 $\{\phi_p(\boldsymbol{x})\}$ (where $\boldsymbol{x} \equiv \{\boldsymbol{r}, \sigma\}$ 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 the basic information for fully characterizing 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 [156]. The $p$-benzyne 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 file configuration shown in Table VI (see Section 19 in the supplementary material of Ref. [299]).

| | | | |
|---|---|---|---|
| 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 VI: Molecular geometry of $p$-benzyne in ångströms, in terms of the xyz file format; see Ref. [299].

In addition to the molecule specifications, we need to select a basis set $\{\phi_p(\boldsymbol{x})\}$. 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 `STO-nG` 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 that 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 both space and 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`) [156]. 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 considering only 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 (see Ref. [300]). For example, for the $p$-benzyne molecule, the full active space involves 68 active spin-orbitals for the `STO-3G` basis and 124 active spin-orbitals for the `6-31G` basis, and the frozen-core approximation involves 56 active spin-orbitals for the `STO-3G` basis and 112 active spin-orbitals for the `6-31G` basis, 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 the basis set and the active space have been selected, we can generate the associated second quantized Hamiltonian, as given in Equation (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 [154] or the Bravyi–Kitaev [155] transformation, to obtain the Hamiltonian in the Pauli-product form. For a Hamiltonian acting on $n$ qubits, it can be expressed as

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

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 logic circuits in the Trotter-based approaches to QPE and how we can guarantee electronic-structure quantum computations to estimate the Hamiltonian eigenvalues within a given target accuracy by satisfying certain error bounds, which in turn must be within given *error budgets*. Our analysis of Trotter errors and their propagation into phase estimation closely follows the approach of Ref. [35]. The propagation of errors in qubitization is discussed in the next subsection.

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 $\theta_\ell$ related to the coefficient $\gamma_\ell$ in the Hamiltonian in Equation (A3)). The latter is approximated by some sequence of the form $HTHST^\dagger \ldots HS$. This approximation is found using circuit synthesis tools based on either the Solovay–Kitaev (SK) algorithm or the Ross–Selinger (RS) algorithm [170, 171] 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 [297]

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

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

Using Propositions 9 and 10 from Ref. [297], we obtain the following analytic error bounds in terms of the spectral operator norm:

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

$$\|\mathcal{S}_2(t) - \exp(-itH)\| \leq \frac{t^3}{12}\sum_{l_1=1}^{L}\left\|\left[\sum_{l_3=l_1+1}^{L} H_{l_3}, \left[\sum_{l_2=l_1+1}^{L} H_{l_2}, H_{l_1}\right]\right]\right\| + \frac{t^3}{24}\sum_{l_1=1}^{L}\left\|\left[H_{l_1}, \left[H_{l_1}, \sum_{l_2=l_1+1}^{L} H_{l_2}\right]\right]\right\|$$

$$\leq \frac{t^3}{3}\sum_{l_1=1}^{L}\sum_{l_3=l_1+1}^{L}\sum_{l_2=l_1+1}^{L} C_{l_1 l_2 l_3}|\gamma_{l_1}\gamma_{l_2}\gamma_{l_3}| + \frac{t^3}{8}\sum_{l_1=1}^{L}\sum_{l_2=l_1+1}^{L} C_{l_1 l_2}|\gamma_{l_1}^2\gamma_{l_2}|, \tag{A7}$$

where $C_{lj} = 1$ if $\left[P^{(j)}, P^{(l)}\right] \neq 0$, and $C_{lj} = 0$ if $\left[P^{(j)}, P^{(l)}\right] = 0$; similarly, $C_{l_1 l_2 l_3} = 1$ if $\left[P^{(l_3)}, \left[P^{(l_2)}, P^{(l_1)}\right]\right] \neq 0$, 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 [297].

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

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

The smaller the time $\tau := t/r$ for a single Trotter slice is, the better the approximation associated with Trotterization becomes. Note that $\lim_{r\to\infty}[\mathcal{S}_p(t/r)]^r = \exp(-itH)$. 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 $HTHST^\dagger \ldots HS$ 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 $\{H, S, T, \text{Pauli gates}, \text{CNOT}\}$ after running the SK algorithm and using other circuit synthesis tools, and let $\Delta_{\text{synth}}$ 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 $\delta$, to be a given upper bound on the allowable circuit synthesis error:

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

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

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

$$\tilde{\mathcal{S}}_1(t) := \prod_{\ell=1}^{L} \tilde{U}_\ell(t),$$

$$\tilde{\mathcal{S}}_2(t) := \left(\prod_{\ell=L}^{1} \tilde{U}_\ell(t)\right)\left(\prod_{\ell=1}^{L} \tilde{U}_\ell(t)\right). \tag{A10}$$

The operator logarithm is well-defined, because $\left[\tilde{\mathcal{S}}_p(\tau)\right]^r$, 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 $\widetilde{\mathcal{U}}(t) := \exp\left(-it\tilde{H}_{\text{eff}}(t)\right)$

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 $\widetilde{\mathcal{U}}(t)$, that is, the quantum circuit implementation of the QPE algorithm is designed to estimate the energy eigenvalues of the effective Hamiltonian $\tilde{H}_{\mathrm{eff}}(t)$ (rather than those of $H$). Since $\widetilde{\mathcal{U}}(t)$ 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 $\tilde{H}_{\mathrm{eff}}(t)$.

We aim to bound $|E_0 - E_{\mathrm{eff}(t),0}|$, which is the difference between the ground-state energy $E_0$ of $H$ (which we aim to estimate) and the lowest eigenvalue of $\tilde{H}_{\mathrm{eff}}(t)$ (which we actually estimate). According to Lemma 3 in the supplementary material of Ref. [35], for any given target error bound $\epsilon$, $\|H - H_{\mathrm{eff}}(t)\| \leq \epsilon$ also implies $|E_0 - E_{\mathrm{eff}(t),0}| \leq \epsilon$. Moreover, according to Lemma 4 in Ref. [35], the assumption that $\|\exp(-itH) - \exp(-itH_{\mathrm{eff}}(t))\| \leq \gamma(t)\,t$ is true for some nondecreasing continuous function $\gamma(t)$ on $[0,\infty)$ implies $\|H - H_{\mathrm{eff}}(t)\| \leq \gamma(t)$. 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:

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

where

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

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

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

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

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

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

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

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

Similar to the approach used in Ref. [35], we define $\epsilon_1 := \Delta E_{\mathrm{TS}[p]}$, $\epsilon_2 := rL\Delta_{\mathrm{synth}}/t$ or $\epsilon_2 := r(2L-1)\Delta_{\mathrm{synth}}/t$ depending on whether we use the first-order or second-order PF, and $\epsilon_3$ 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 $\epsilon_1$, $\epsilon_2$, and $\epsilon_3$ 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,

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

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 $\|\exp\left(-itH\right) - \left[\mathcal{S}_2\left(\tau\right)\right]^r\|$ via full numerical computations for $p$-benzyne Hamiltonians pertaining to small active spaces HL$\pm n$ for $n = 0, 1, 2$. We inferred the corresponding required evolution time $\tau$ 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 $\beta$ 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 [301]. Based on the deduced scaling, we have estimated $\beta$ values for larger active spaces via regression.

### 3. Propagation of errors in qubitization

In this section, we discuss the various errors incurred in the process of generating the logical quantum circuits for the double-factorized (DF) qubitization algorithm, and how we can guarantee electronic-structure quantum computations to estimate the Hamiltonian eigenvalues to a given target accuracy by satisfying certain error bounds within predetermined error budgets. In QPE based on qubitization, the spectrum of the molecular Hamiltonian

$H$ is sampled by measuring the phase that is accumulated on an initial input quantum state through multiple controlled executions of the quantum walk operator $e^{i \arccos(H/\lambda)}$ (or $e^{i \arcsin(H/\lambda)}$) in place of the time-evolution operator $\exp(-iHt)$, see Refs. [166, 167]. In the DF qubitization algorithm, there are four main sources of error (see Refs. [150, 160]):

1. Error associated with the truncation in the double-factorization procedure;

2. Error associated with the binary approximation of the Hamiltonian coefficients when implementing the PREPARE oracle;

3. Error associated with the approximation of the individual Givens rotations in implementing the basis rotations; and

4. Error associated with the measurement readout in phase estimation.

The first source of error arises from truncating the small eigenvalues of the double-factorized Hamiltonian, but by keeping track of the values of the truncated eigenvalues it is possible to bound the 2-norm of the difference between the original Hamiltonian and the truncated Hamiltonian. The second and third sources of error arise from approximations in the implementation of the LCU oracles associated with the qubitization operator, namely, binary approximations of the coefficients involved in the PREPARE operation as well as binary approximations of the rotation angles in the diagonalization operations involved in the innermost decomposition of the double-factorized Hamiltonian (with finite bits of precision). The final contribution to the error is the imprecision in the measurement readout of QPE; this error and the requisite number of logical ancillary qubits can be bounded using the total number of repetitions of the qubitization operator. For chemical significance, the total overall target error in estimating the ground-state energy should be at most 1.6 millihartrees, that is, the overall error budget for estimating the lowest eigenvalue of the molecular Hamiltonian is $\epsilon = 1.6$ mHa (or the much lower target error $\epsilon = 0.1$ mHa for quantitative accuracy).

At the logical level, it is possible to break down the error in QPE of the qubitization approach as in Equation (23) of Ref. [77], namely, the output energy of the QPE is within

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

of the Hamiltonian used in the qubitization approach. Here, $\lambda$ is the 1-norm of the Hamiltonian (computed as the sum of the absolute values of the Hamiltonian weightings), $m$ is the number of bits in the QPE, $\epsilon_{\text{QFT}}$ is related to the error in implementing the quantum Fourier transform and is usually negligable, and $\epsilon_H$ is the systematic error in the phase resulting from the gate synthesis in

implementing the qubitization operator of the double-factorized Hamiltonian. According to Equation (A19), if we aim to estimate spectra to within error $\Delta E$, we can achieve this by choosing (see [77]):

$$m = \left\lceil \log_2\left(\frac{\pi\lambda}{\sqrt{2}\Delta E}\right) \right\rceil, \; \epsilon_H \leq \frac{\sqrt{2}\Delta E}{4\lambda}, \; \epsilon_{\mathrm{QFT}} \leq \frac{\sqrt{2}\Delta E}{4\pi\lambda}.$$
(A20)

Following the analysis outlined in the Appendix of Ref. [77], we can bound the error contributions arising from finite-bit precision approximations of the quantum circuit encoding of the double-factorized Hamiltonian. Let $H = \sum_{\ell=0}^{L-1} w_\ell H_\ell$ denote the double-factorized Hamiltonian (after truncation of the terms pertaining to small eigenvalues), and let $\widetilde{H} = \sum_{\ell=0}^{L-1} \widetilde{w}_\ell \widetilde{H}_\ell$ denote the corresponding approximate encoding of $H$ resulting from finite-bit precision approximations of Hamiltonian coefficients and Givens rotations. Then, the associated propagated error into the eigenphase obeys:

$$\begin{aligned}
\epsilon_H &\leq \|e^{i\arccos(H/\lambda)} - e^{i\arccos(\widetilde{H}/\lambda)}\| \\
&\leq \|\arccos(H/\lambda) - \arccos(\widetilde{H}/\lambda)\| \\
&\leq \sum_{p=0}^{\infty} \frac{(2p-1)!!}{\lambda^{2p+1}(2p+1)(2p)!!}\|H^{2p+1} - \widetilde{H}^{2p+1}\|. \quad (A21)
\end{aligned}$$

Let $\chi := \|H - \widetilde{H}\|$. Then the term $\|H^{2p+1} - \widetilde{H}^{2p+1}\|$ can be bounded from above by $(2p+1)(\|H\| + \chi)^{2p}\chi$, which yields

$$\begin{aligned}
\epsilon_H &\leq \sum_{p=0}^{\infty} \frac{(2p-1)!!}{\lambda^{2p+1}(2p)!!}(\|H\| + \chi)^{2p}\chi \\
&\leq \frac{\chi}{\lambda}\left[1 - \left(\frac{\|H\| + \chi}{\lambda}\right)^2\right]^{-1/2}. \quad (A22)
\end{aligned}$$

Thus, in order for the error contributions arising from finite-bit precision approximations in implementing the qubitization quantum circuit to satisfy the bound specified in Equation (A20), we must choose the bit precisions to be large enough such that

$$\chi \leq \frac{\sqrt{2}\Delta E}{4\left(1 + \frac{\Delta E^2}{8\lambda^2}\right)}\left(1 - \|H\|^2/\lambda^2\right). \quad (A23)$$

This inequality is used to determine the number of bits required for the binary approximations of the Hamiltonian coefficients $w_\ell$ and the binary approximations of the rotation angles associated with the diagonalizing fermionic basis transformations that yield the double-factorized form.

## Appendix B: Quantum resource estimation 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 [67]. The software's approach to creating a fault-tolerant architecture is described in Ref. [62]. For a given quantum circuit and success probability, we generate an architecture that would feasibly run the computation at the requisite precision. This architecture 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, 61, 63]. 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 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 [63]. 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. [63], 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 logic 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 Figure 15, 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 [128, 129] as described in Section 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 [61]. 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. [63] 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 $d_{\text{core}}$ parity checks, where $d_{\text{core}}$ 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, $E_{\text{core}}$, and in the distillation of magic states in the MSF, $E_{\text{MSF}}$. Therefore,

$$E_{\text{core}} + E_{\text{msf}} \leq E. \tag{B1}$$

The errors of the core processor and the MSF are modelled and predicted following the pipeline described in Ref. [62]. 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

$$E_{\text{core}} \approx (2Q + \sqrt{8Q} + 29)T e_{\text{mem,core}}, \tag{B2}$$

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 15, 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 [132] 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 Ref. [62], considering $e_{\text{prep}}$ 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 $l \in \{1, \ldots, L\}$:

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

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

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

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

$$E_{\text{msf}} = e_{\text{core}}T. \tag{B3}$$

The choice of hardware parameters to determine the required number of distillation levels $L$ and code distances $d_l, \forall l \in 1, \ldots, L+1$, 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. [62]. 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, that is, $E_{\text{msf}} \leq E - E_{\text{core}}$. Next, it determines the number of distillation levels $L$ required to meet the magic state error rate requirement $e_{\text{core}}$ derived from Equation (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 [62].

## 3. Comparison with the AzureQRE toolkit

To provide additional logical and physical resource estimates, we use the Azure Quantum Resource Estimator (AzureQRE) [145, 302]. 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. [302] and [61] for full details about the architectural assumptions, but briefly highlight Azure-QRE 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 VII, where we have also included the results from TopQAD's resource estimates for the DF qubitization algorithm in Table V 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 [62] 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

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

where $h_{ij}$ and $h_{ijkl}$ are the one- and two-electron integrals, $\sigma$ and $\rho$ 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 [303].

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 [304]:

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

Each matrix $\mathcal{L}^{(\ell)}$ can then be be further decomposed, giving a set of eigenvalues $\{\lambda_m^{(\ell)}\}$ and a diagonalizing uni-

| | Target error $\epsilon$ | $N_{\mathrm{orb}}$ | Baseline Parameter Set | | | Target Parameter Set | | | Desired Parameter Set | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | # Phys. qubits | Phys. time | QEC code distances | # Phys. qubits | Phys. time | QEC code distances | # Phys. qubits | Phys. time | QEC code distances |
| **AzureQRE** | 1.6 mHa | 6 | - | - | - | $1.1\times10^7$ | 17.4 minutes | 75 | $3.1\times10^6$ | 10.0 minutes | 43 |
| | | 18 | - | - | - | $2.3\times10^7$ | 16.2 hours | 89 | $7.6\times10^6$ | 9.3 hours | 51 |
| | | 26 | - | - | - | $3.1\times10^7$ | 2.8 days | 93 | $9.9\times10^6$ | 1.6 days | 53 |
| | | 76 | - | - | - | $9.7\times10^7$ | 286.3 days | 109 | $3.0\times10^7$ | 160.2 days | 61 |
| | 0.1 mHa | 6 | - | - | - | $1.6\times10^7$ | 6.1 hours | 85 | $5.0\times10^6$ | 3.4 hours | 47 |
| | | 18 | - | - | - | $3.0\times10^7$ | 15.0 days | 97 | $9.4\times10^6$ | 8.5 days | 55 |
| | | 26 | - | - | - | $4.1\times10^7$ | 62.1 days | 103 | $1.3\times10^7$ | 34.4 days | 57 |
| | | 76 | - | - | - | $1.3\times10^8$ | 21.0 years | 119 | $4.0\times10^7$ | 11.8 years | 67 |
| **TopQAD** | 1.6 mHa | 6 | $1.4\times10^7$ | 3.3 minutes | 25, 59 \| 73 | $1.3\times10^6$ | 1.2 minutes | 23 \| 27 | $8.2\times10^5$ | 35.8 seconds | 17 \| 21 |
| | | 18 | $4.8\times10^7$ | 1.4 days | 15, 31, 75 \| 93 | $3.6\times10^6$ | 7.3 hours | 11, 27 \| 33 | $1.7\times10^6$ | 5.6 hours | 21 \| 25 |
| | | 26 | $5.5\times10^7$ | 5.4 days | 15, 33, 79 \| 95 | $4.7\times10^6$ | 1.3 days | 11, 29 \| 35 | $2.6\times10^6$ | 23.6 hours | 23 \| 27 |
| | | 76 | $1.2\times10^8$ | 1.4 years | 17, 37, 89 \| 107 | $1.3\times10^7$ | 121.8 days | 13, 33 \| 39 | $7.7\times10^6$ | 96.8 days | 25 \| 31 |
| | 0.1 mHa | 6 | $2.3\times10^7$ | 12.6 hours | 29, 77 \| 91 | $3.0\times10^6$ | 2.7 hours | 11, 27 \| 31 | $1.4\times10^6$ | 2.2 hours | 21 \| 25 |
| | | 18 | $5.8\times10^7$ | 30.8 days | 15, 35, 91 \| 105 | $5.1\times10^6$ | 6.5 days | 13, 31 \| 35 | $2.6\times10^6$ | 5.4 days | 23 \| 29 |
| | | 26 | $7.9\times10^7$ | 132.1 days | 19, 35, 85 \| 115 | $6.8\times10^6$ | 28.5 days | 13, 31 \| 39 | $3.4\times10^6$ | 21.2 days | 25 \| 29 |
| | | 76 | $1.5\times10^8$ | 28.5 years | 17, 41, 99 \| 119 | $1.8\times10^7$ | 6.5 years | 15, 35 \| 43 | $9.8\times10^6$ | 5.0 years | 29 \| 33 |

Table VII: Physical resource estimates generated by TopQAD [67] and AzureQRE [145] for implementing the QPE algorithm on electronic-structure quantum circuits associated with the $p$-benzyne and FeMoco molecules, for two precisions in energy estimation: qualitatively accurate computation within a target error 1.6 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 $N_{\mathrm{orb}}$. The data for $N_{\mathrm{orb}} = 6, 18, 26$ correspond to active space selections HL$\pm 2, 8, 12$ (using HL$\pm n$ to denote "HOMO$-n$ and LUMO$+n$"; see Appendix A 1 for an explanation of these terms) for $p$-benzyne using the `6-31G` basis to represent the fermionic orbitals; the data for $N_{\mathrm{orb}} = 76$ 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, as summarized in Table I. Note that the symbol "-" represents that AzureQRE estimates that the baseline parameter set is above the QEC threshold.

tary $U^{(\ell)}$. This leads to the double-factorized form of the Hamiltonian $H_{\mathrm{DF}}$:

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

(C3)

where $\tilde{h}_{ij} \equiv h_{ij} - \frac{1}{2} \sum_l h_{illj}$ comes from the reordering of the creation and annihilation operators. By truncating some of the eigenvalues, a low–rank approximation can be obtained. There exist efficient walk operators, $W$, which implement this Hamiltonian as a quantum circuit, as described in Ref. [303].

## 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) [173] and density matrix renormalization group (DMRG) [163]. For the FCI calculations, we note that Ref. [173] 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 ($\mathcal{O}(N_{\mathrm{det}}^2)$) 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 $N_{\mathrm{det}} = \left( N_o! / (N_o - N_e)! N_e! \right)^2$ and, assuming a factor of 1000 in parallelism, compute the time for various numbers of orbitals, consistent with the 512 CPUs used in Ref. [173].

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 $\chi$ with number of orbitals for generic quantum chemistry problems, but it is generally expected to scale exponentially for strongly

correlated systems. Estimates of the necessary bond dimension for various homogeneous catalysts are reported in Ref. [163], as well as runtimes for smaller bond dimension DMRG calculations. Using the data in Table 3 of Ref. [163], 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. [163].

## 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 values of hardware noise parameters, such as those provided in Table I.

Imperfect gates are modelled 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,

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

where $n$ is the number of gate qubits.

Idle qubits are succeptible to errors 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

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

The errors in state preparation and reset are captured by assuming that with rate $p$ the orthogonal state is produced, that is, $|0\rangle$ 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

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

from which we directly determine the rate $p = 1 - F_{\mathrm{SPAM}}$.

## Appendix F: High-performance real-time control and decoding with DGX Quantum

Nvidia DGX Quantum [113], developed in collaboration with Quantum Machines, is a groundbreaking architecture that tightly integrates advanced classical and quantum computational capabilities. It combines Nvidia Grace Hopper superchips with the Quantum Machines OPX1000 control system, achieving microsecond-scale latency between GPU and QPUs. This connection enables latency-critical work like QEC to be accelerated with GPUs. The system is designed to scale with both quantum and classical computing demands; as quantum computers advance, additional DGX Quantum nodes can be incorporated to interface with more QPUs. Moreover, these nodes can be connected by both classical and quantum interconnects to scale up to large accelerated quantum supercomputing systems, as shown in Figure 3. DGX Quantum is also QPU agnostic, enabling all qubit modalities to be integrated within the architecture.

The Nvidia GH200 superchip is a key part of the DGX Quantum integrated system, delivering exceptional performance for hybrid quantum–classical workloads through its heterogeneous architecture. This architecture eliminates bottlenecks in data movement and ensures optimal performance for latency-sensitive quantum–classical interactions, as required in QEC and hybrid algorithm execution. Advanced features of the Hopper GPU, including asynchronous execution engines and thread block reconfiguration, further enhance computational efficiency, ensuring that all resources are fully utilized in the DGX Quantum integrated system. Future DGX Quantum systems may use Grace Blackwell GB200 superchips (see Figure 46). A comparison of Grace Hopper and Blackwell superchips is shown in Table VIII, which demonstrates the GB200 superchip's significantly improved specifications over those of the GH200. It is an important example of improvement because to calculate the future true costs of quantum computing, one needs to factor in the decreasing cost of error correction due to improvement in classical computing infrastructure.

DGX Quantum also enables many other control mechanisms, such as error mitigation, dynamic circuit programming, and adaptive circuit knitting. By directly writing quantum control data into the GH200's local memory, latency overhead is eliminated, ensuring that the system can respond to quantum hardware feedback in real time. This feature is particularly valuable for iterative classical–quantum applications. For less latency-critical scenarios, users can queue quantum programs locally and remotely, enabling resource flexibility while maintaining access to DGX Quantum's high-performance computational capabilities.

| | GH200 Grace Hopper Superchip | GB200 Grace Blackwell Superchip |
|---|---|---|
| **Configuration** | 1 Grace CPU : 1 Hopper GPU | 1 Grace CPU : 2 Blackwell GPUs |
| **FP4 Tensor Core** | 20 PFLOPS | 40 PFLOPS |
| **FP8/FP6 Tensor Core** | 4 PFLOPS | 20 PFLOPS |
| **INT8 Tensor Core** | 4 POPS | 20 POPS |
| **FP16/BF16 Tensor Core** | 2 PFLOPS | 10 PFLOPS |
| **TF32 Tensor Core** | 1 PFLOPS | 5 PFLOPS |
| **FP32** | 67 TFLOPS | 180 TFLOPS |
| **FP64** | 34 TFLOPS | 90 TFLOPS |
| **FP64 Tensor Core** | 67 TFLOPS | 90 TFLOPS |
| **GPU Memory \| Bandwidth** | 96 GB HBM3 \| 4 TB/s | Up to 384 GB HBM3e \| 16 TB/s |
| **NVLink Bandwidth** | 0.9 TB/s | 3.6 TB/s |
| **CPU Core Count** | 72 Arm Neoverse V2 cores | 72 Arm Neoverse V2 cores |
| **CPU Memory \| Bandwidth** | Up to 480GB LPDDR5X \| Up to 512 GB/s | Up to 480GB LPDDR5X \| Up to 512 GB/s |

Table VIII: Grace Hopper vs. Grace Blackwell superchip specifications.



Figure 46: Picture of the Nvidia Grace-Blackwell GB200 superchip. The GB200 contains two Nvidia Blackwell B200 Tensor Core GPUs with an Nvidia Grace CPU, interconnected via a 900 GB/s NVLink-C2C interface, creating a unified memory architecture. This configuration provides a total of 896 GB of memory, combining 384 GB of HBM3e memory on the GPUs and 512 GB of LPDDR5 memory on the CPU.

### Appendix G: CUDA-Q: A software platform for heterogeneous quantum–classical computing

DGX Quantum integrated systems are built to run Nvidia CUDA-Q [114], which is an open source programming framework and compiler toolchain designed as an efficient and unified platform for programming hybrid quantum–classical systems. CUDA-Q supports both Python and C++ programming interfaces. Its kernel-based programming capabilities are specifically tailored for hybrid quantum–classical systems. CUDA-Q programs use the concept of a quantum kernel to differentiate between host and quantum device code, with kernels specifying a target for compilation and execution. The CUDA-Q platform also includes the NVQ++ compiler, which supports split compilation by lowering quantum kernels to a multi-level intermediate representation (MLIR) and a quantum intermediate representation (QIR). This ensures the seamless integration of classical and quantum resources needed for accelerating applications in large-scale quantum computing.

CUDA-Q includes specialized libraries designed to streamline the development of quantum algorithms and support advanced research. The CUDA-Q QEC library provides optimized implementations of key quantum error correction primitives (such as GPU-accelerated decoders) and the CUDA-Q Solvers library includes application development primitives (such as prebuilt optimized kernels for the VQE, ADAPT-VQE, and QAOA methods). CUDA-Q also offers GPU-accelerated quantum dynamics simulations, enabling accurate modeling of open quantum systems—a critical capability for the design, characterization, optimization, and scaling of QPUs.

Another key advantage of CUDA-Q is its ability to scale the performance of quantum simulations. Leveraging Nvidia's multi-GPU acceleration, CUDA-Q can achieve a massive speedup over traditional CPU-based simulations [114], enabling researchers to efficiently simulate large quantum circuits. CUDA-Q supports multiple simulation technique backends such as state vector and tensor networks, allowing users to select from among various simulation methods for their specific problem.

Additionally, CUDA-Q is interoperable with AI software and the CUDA software ecosystem. This ecosystem is a comprehensive suite of tools, libraries, and frameworks designed to harness the computational power of GPUs for diverse applications, ranging from AI and data science to HPC and quantum simulation. It includes components that provide developers with optimized building blocks, scalable programming models, and integration capabilities for hybrid quantum–classical workflows.