VelociTI: An Architecture-level Performance Modeling Framework for Trapped Ion Quantum Computers

Alexander Hankin[1,3] Abdulrahman Mahmoud[1] Mark Hempstead[2]
David Brooks[1] Gu-Yeon Wei[1]

[1]*Harvard University, Cambridge, MA, USA*
[2]*Tufts University, Medford, MA, USA*
[3]*Intel Labs, Hillsboro, OR, USA*
correspondence: alexander.hankin@intel.com

Abstract

Trapped-ion (TI) qubit architectures have recently become a promising candidate for designing and building quantum computers. In the current noisy-intermediate scale quantum (NISQ) era, TI qubits stand out for their connectivity and reliability over other candidates such as superconducting qubits. However, physical constraints stemming from fine-grained frequency control of TI qubits introduce limitations to the maximum number of trapped-ions in a quantum computing system. This fundamentally challenges the design of large TI-based quantum computers, with various quantum applications requiring a large number of qubits for practical realization.

Recent work has proposed TI Quantum Charge Coupled Devices (QCCD) which provides mechanisms to link multiple ion-chains together to address the issue of scalability. While such advances help increase the total qubit count in a TI system, the weak links between ion chains introduce a performance bottleneck and gate-latency penalty. Prior TI modeling toolflows have not explored the performance and scalability implications introduced by weak links on the design of future TI systems; in this work, we directly elevate the weak link as an architectural knob, and present an architecture-level performance modeling framework called VelociTI. We use VelociTI to study the performance trade-offs in a trapped-ion quantum computing design and find that optimal scheduling of qubits can provide a 6.2× speedup in performance.

1. Introduction

Quantum computer (QC) designs have advanced rapidly over the last decade. Google [4], IBM [9], and IonQ [13] have all recently launched 100+ qubit processors, up from single digits at the turn of the century. Multiple quantum computing simulators have also been introduced by various cloud vendors, including IBM (Qiskit [1]), Microsoft (Azure Quantum [5]), and Amazon (Braket [2]), allowing researchers to begin exploring and studying quantum applications and future hardware designs.

Despite their name, quantum computers are more akin to quantum processing units (QPUs), and are better construed as extremely fast and efficient accelerators for solving certain problems that classical computers may struggle with [3, 8, 11, 25]. In the modern age of heterogeneous system designs, such a computing model would require that classical computers set up the inputs for a QPU, then the use of quantum acceleration for the task at hand, followed by classical postprocessing. While various costs may be associated with the setup, execution, and collection of results in this workflow, the computational acceleration provided by a QPU is expected to be sufficiently large, such that associated overheads are comparatively justified. Thus, under this premise, most recent work has focused more on *functionality* by addressing quantum decoherence (i.e., error) and *scaling up* quantum computers (in order to eventually solve useful problems), with less attention being attributed to actual QPU *performance*.

While scale and accuracy are fundamental tenants for practical QPU realization in the noisy intermediate-scale quantum (NISQ) era, the tenant of performance is an important metric for evaluating and comparing different *hardware solutions* for a QPU. In particular, there are a couple of competing technologies for qubit implementations, each with their strengths and weaknesses. In this work, we focus on TI technology for qubit implementation, and take an architecture-directed approach for evaluating the performance and scaling potential of QPUs.

The building block of the TI QC is referred to as a chain. This chain contains the trapped ions which act as the qubits which are used for computation. Multiple published systems can address up to 32 ions through the use of a 32-channel acousto-optic modulator (AOM) [10, 12, 18]. To scale the number of qubits, chains are linked together to form multiple ion chains. TI QCs possess the advantage of providing all-to-all connectivity between qubits within a chain, as well as the condition that the quality of all qubit pairs in a chain are equal. The drawback is connection latency between multiple chains: communication across chains has a longer latency than operations within a chain, due to the optics required over free-space paths which introduces significant drift and noise into the system [21, 24, 28]. We refer to such connections between chains as *weak links*.

Weak links present an architecture-level knob for scaling TI-based QPUs. While weak links provide physical mechanisms to scale qubits beyond their chain limitations,

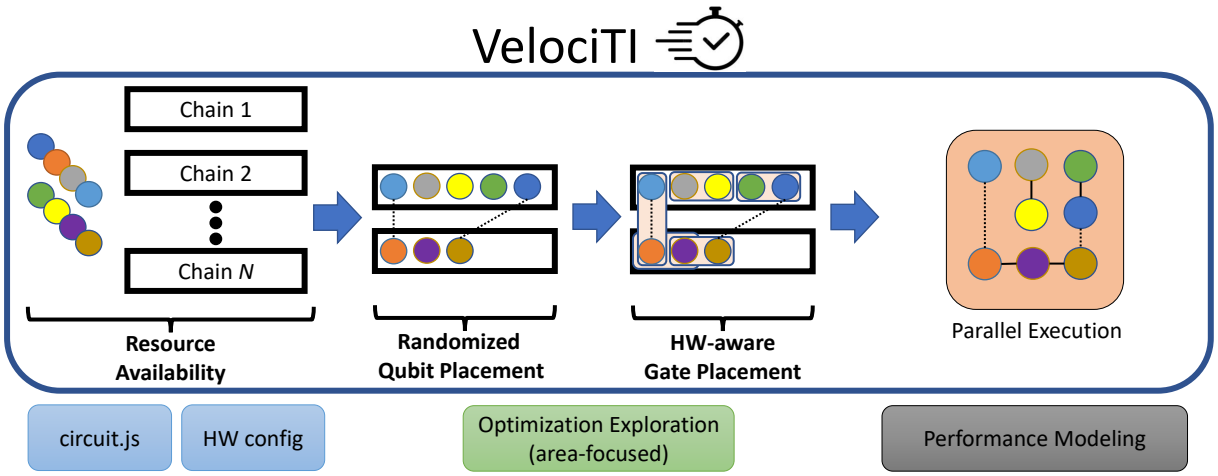

Figure 1: **High-level overview of VelociTI**.

exploring the trade-off between "horizontal" and "vertical" scaling has not been performed before. Additionally, understanding the impact of weak links on performance can help usher and navigate early design space explorations for QPU hardware, elevating hardware-specific details for architectural evaluation. To the best of our knowledge, we are the first work to provide architecture-level abstractions and performance models to evaluate the weak link as TI QCs scale from the NISQ-era systems (50-100 qubits) to beyond. We propose VelociTI: an architecture-level performance modeling framework for Trapped-Ion quantum computers. We make the following novel contributions:

- An architecture-level performance modeling framework with TI abstractions for quantum and classical architects to evaluate, optimize, and scale NISQ-era QCCD-based TI system designs (§2, §3)
- An evaluation of optimal TI configuration for a range of quantum applications on QCCD-based TI systems (§4.1)

2. VelociTI Framework Overview

While the differences between classical and quantum accelerators may seem stark, this also creates a unique opportunity for computer architects to bridge the gap between the hardware and the software in the quantum space, and explore the integration of a QPU within a classical computing system.

2.1. Performance Modeling Goals

The realm of quantum computing offers a huge design space for hardware and software exploration. Understanding and exposing the correct abstractions from the physical qubit level will greatly benefit architects and compiler designers to bridge the gap for algorithm and software developers. At the same time, architects can help understand the needs of software, using the knowledge to explore and direct the qubit technology for the NISQ era. One of the primary goals of VelociTI is to bridge the gap between low-level hardware implementations (i.e., of the trapped-ion chain) and higher system level challenges such as scalability and performance. While many prior works in the domain have abstracted down to the gate level only (since superconducting qubits have been the only practical technology for QCs for many years), our framework helps expose architects to the unique advantages of trapped-ion technology.

Such an exploration requires modeling and simulation workflows to gauge the benefits and drawbacks of the technology, particularly as physicists tackle the challenging aspect of controlling quantum properties for computation. Prior work has focused on functional correctness of computation on TI quantum computers [20, 27], addressing the issues of error accumulation and information management. However, the issue of *scale* has been a challenge, both with horizontal scaling techniques via longer ion-chain studies [23] and vertical scaling via weak links.

2.2. VelociTI Design and Workflow

Figure 1 depicts a high level overview of VelociTI. VelociTI is broadly composed of three stages:

1) **Boundary conditions:** The input into the system is a set of boundary conditions required for simulation. This is composed of two parts: a general circuit description, which specifies components that represent a quantum application, such as the number of qubits, the number of 1-qubit gate operations, and the number of 2-qubit gate operations. The second part is a set of timing configurations for the system, including 1-qubit operation latency, 2-qubit operation latency, and the weak-link penalty for communication across chains.
2) **Place-and-route:** Using the boundary conditions as input, the second phase generates potential circuit layouts, in the context of trapped-ion chains and weak-link availability. While the space of possible layouts is extremely large, we focus on an *optimization target* to help generate circuits. In this work, our target is to minimize *area*, although other potential optimizations can be used. Minimizing area implies

understanding whether to place qubits on previously populated chains, or to introduce a new chain. Additionally, gate operations have constraints that need to be managed as well in this step, as 2-qubit gates can not necessarily operate on any two qubits in the system (only within a chain or at the weak links).

3) **Performance Modeling:** The third phase uses the generated circuit layouts to predict total execution time. As quantum computers offer a large degree of computational parallelism, our models take into account circuit design to find the shortest execution time possible, and compare to a baseline implementation where no parallelism is enforced (which may occur if a system is naively scheduled).

Finding an optimal placement of qubits and gates within a set of trapped-ion chains is a prohibitively expensive operation. Instead, VelociTI uses a pseudo-random placement policy for generating a single circuit layout, followed by averaging across multiple circuit designs during the performance modeling.

We first compute a minimal number of ion-chains, as a function of the ion-chain length and the qubits present in the system (provided from the boundary conditions). With the number of ion-chains fixed, we randomly place qubits and distribute them across the chains. Subsequently, we place 1-qubit and 2-qubit gates to operate on the qubits, with the condition that 2-qubit operations are restricted to intra-chain operations or weak-link operations - in other words, communication between two chains via a gate *must* occur via the weak link connection, and only the qubits on the edge of a weak link can be used for such communications.

3. Trapped-Ion Chain Performance Modeling

In this section, we outline our performance models for a multi-chain TI quantum computer architecture. We first implement a serial performance model as a baseline study, followed by a parallel model for capturing the parallel nature of a TI quantum computer.

3.1. Model parameters

The model parameters which VelociTI uses to represent a quantum TI system includes the total number of qubits, the total number of gates, the latency required for different kinds of gates (all 1-qubit gates have the same latency and all 2-qubits have the same latency except for if there is a weak link involved [6, 26, 28]), the penalty term for a 2-qubit gate latency that involves a weak link, and the optimization target (e.g. area) for determining the number of ion chains to be used. The computed parameters include the number of chains, the number of available weak links, and the number of weak links that were used during gate placement.

3.2. Intra-chain parallelism using a directed graph

With the TI architecture, it is possible for chains to operate in parallel if a weak link is not involved. This intra-chain parallelism allows for higher performance and opens the door to more optimization opportunities. Conceptually, two chains can operate in parallel so long as scheduled gate operations do not cross the weak-link boundary, at which point serialization operations are required for ordering. To allow us to take advantage of compute parallelism, we use a directed graph for gates to compute longest paths. This ultimately will be used in calculating performance for a circuit.

Nodes are used to represent *gate* operations, introducing an explicit ordering which can be used to extract parallelism. Specifically, the directed edges between nodes indicates the order of operations. Thus, there will be multiple paths in a graph to represent the different communication paths between qubits in a circuit. Edge weights are used to represent the latency of a node/gate. Given that an edge connects two nodes/gates, we set an edge weight to correspond to an incoming gate's latency. To make sure that all gate latencies are accounted for, in the case of an edge involving a "start node", the corresponding edge weight will be the sum of both the incoming and outgoing nodes. A "start node" simply refers to a node which is connected to an input qubit.

3.3. Calculating performance

Once the graph representation is constructed, we can calculate the performance of the circuit. Using the representation that we have built, we can take advantage of existing graph algorithms for computing longest path in order to compute the overall parallel performance of a circuit. The parallel performance model calculates the total latency of each parallel path by summing the edge weights between all nodes in each path, and then returns the highest latency of all the parallel paths. This is the total latency of the circuit.

4. TI Performance Evaluation Using Realistic Circuit Models

To demonstrates the use of VelociTI, we now perform a case study. Given an available, pre-built system with specific parameters defined by the hardware, we determine what is the best mapping of the quantum application onto the system. For our case study, we run VelociTI with 6 quantum computing applications: Supremacy [3, 19], Quantum Approximate Optimization Algorithm (QAOA) [14–16, 22], SquareRoot (Grover's search algorithm [17]), Quantum Fourier Transform (QFT) [7], Adder, and Bernstein-Vazirani (BV) [29]. We choose these applications because they provide a variety in (a) number of qubits, (b) number of 2-qubit gates, and (c) ratio of 2-qubit gates to qubits. An overview of these applications are shown in Table 1. These applications have been used in previously published TI architecture work [23]. For our gate latency model, we use a latency of 1 μs for a 1-qubit gate and a latency of 100 μs for a 2-qubit gate. This gate latency model has been experimentally validated in multiple prior published works on TI systems [6, 26]. In the case of a weak link, we apply a penalty factor of 2 based on experimental validation in previously published work [28].

Application	Qubits	2-qubit Gates
Supremacy	64	560
QAOA	64	1260
SquareRoot	78	1028
QFT	64	4032
Adder	64	545
BV	64	64

TABLE 1: Applications with attributes used in our evaluation.

4.1. Case Study: What is the best estimated performance for a given hardware implementation?

For each application, we run both the serial and parallel model. For this we use a chain length of 16 qubits, and we assume an area optimized architecture where the minimum number of chains are used. Given that the SquareRoot application has more qubits than the others, one additional chain is needed compared to the other applications. For qubit and gate scheduling, we utilize a purely random approach without the use of any optimizations. The resulting performance is shown in Figure 2. We run each application 35 times and each bar shows the average execution time with a vertical bar indicating the maximum and minimum execution time reported for each benchmark. On average, serial execution time is 69.3 ms and parallel performance is 11.2 ms with parallel model achieving an average speedup 6.2× over the serial model. As expected, serial and parallel performance is a function of the number of 2-qubit gates: the application with largest number of 2-qubit gates, QFT, has the longest latency: 403.6 ms serially and 74.5 ms when run with between-chain parallelism enabled.

We also observe that the benefit of using the serial model versus parallel model is benchmark dependent. For Supremacy, QAOA, SquareRoot, QFT, and Adder, parallel speedup is around 6× whereas BV achieves a parallel speedup of 9.9×. This is in part due to the lowest ratio of 2-qubit gates to qubits (1:1) in BV. As the ratio of 2-qubit gates to number of qubits increases, the number of parallel compute paths decreases and the length of the remaining paths increase at an even faster rate as qubits become more connected. This is interesting as there may be applications in the future with an even higher ratio of 2-qubit gates to qubits than the applications used in this case study, in which case—when taking into account the increased overhead of parallel scheduling—it may be more worth it to schedule and execute serially.

We now relax the architecture design to sweep the chain length within a presently achievable range (8, 16, 24, and 32 ions per chain). In the case of Supremacy, QAOA, QFT, Adder, and BV, this creates 8, 4, 3, and 2 weak links, respectively. For SquareRoot which has a larger number of qubits, the number of weak links is 10, 5, 4, and 3, respectively. We look at how the change in chain length affects the performance of the quantum applications. For this analysis, we disregard the serial model as it is consistently worse.

Figure 3 shows parallel execution time for the applications with varying TI chain length. We observe that sweeping chain length from 8 to 32 results in an average speedup of 20%. Again, we notice different behavior for

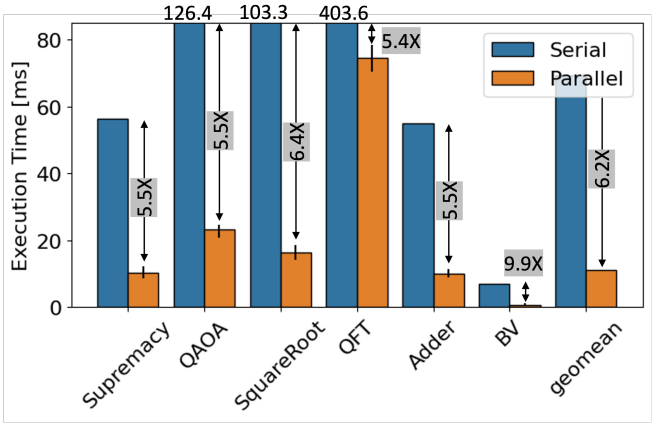

Figure 2: **Estimated performance on a given practical TI hardware implementation**.

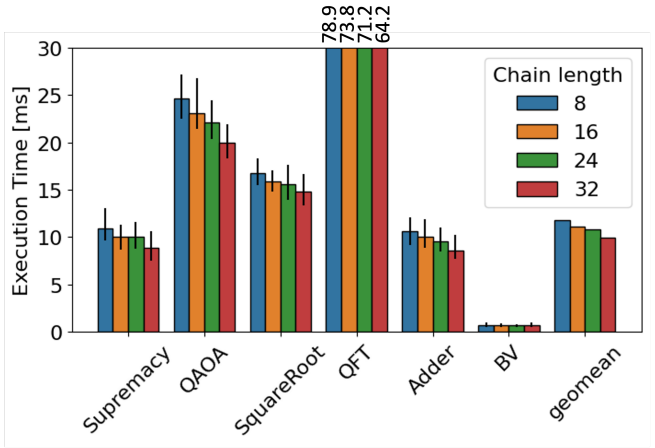

Figure 3: **Estimated performance as a function of chain length.**

BV with a speedup of only 11% between chain length of 8 qubits to 32 qubits. In classical computing, this magnitude of speedup would be significant; however, it is unclear whether such a speedup would matter in a real quantum accelerator use case. Our results show that increasing chain length horizontally suggests a continued improvement in performance for parallel execution. This encourages continued development of longer chains physically.

5. Conclusion

In this work, we developed and concretized architecture-level abstractions for QCCD-based TI systems which we used as a foundation for design and implementing VelociTI, an architecture-level performance modeling framework for Trapped Ion Quantum Computers. We evaluate VelociTI by conducting a case study which represents the use cases of a tool like VelociTI. For a given HW implementation and suite of applications, we find that the place-and-route, i.e. mapping scheme, of the qubits and gates can have a 6.2X difference in performance.

References

[1] G. Aleksandrowicz, T. Alexander, P. Barkoutsos, L. Bello, Y. Ben-Haim, D. Bucherand, F. J. Cabrera-Hernández, J. Carballo-Franquis, A. Chen, C.-F. Chen, J. M. Chow, A. D. Córcoles-Gonzales, A. J. Cross, A. Cross, J. Cruz-Benito, C. Culver, S. González, E. Torre, D. Ding, E. Dumitrescu, I. Duran, A. Eendebak, M. Everitt, I. F. Sertage, A. Frisch, A. Fuhrer, J. Gambetta, B. G. Gago, J. Gomez-Mosquera, D. Greenberg, I. Hamamura, V. Havlicek, J. Hellmers, Herok, H. Horii, S. Hu, T. Imamichi, T. Itoko, A. Javadi-Abhari, N. Kanazawa, A. Karazeev, K. Krsulich, P. Liu, Y. Luh, Y. Maeng, and M. Marques, "Qiskit: An open-source framework for quantum computing," 2019.

[2] Amazon Web Services, "Amazon Braket," 2020. [Online]. Available: https://aws.amazon.com/braket/

[3] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, Oct. 2019.

[4] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[5] Azure, "Azure Quantum," 2022. [Online]. Available: https://azure.microsoft.com/en-us/services/quantum/

[6] C. J. Ballance, T. P. Harty, N. M. Linke, M. A. Sepiol, and D. M. Lucas, "High-fidelity quantum logic gates using trapped-ion hyperfine qubits," *Phys. Rev. Lett.*, vol. 117, p. 060504, Aug 2016.

[7] E. Bernstein and U. Vazirani, "Quantum complexity theory," in *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, ser. STOC '93. New York, NY, USA: Association for Computing Machinery, 1993, p. 11–20.

[8] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, Sep. 2017.

[9] J. Chow, O. Dial, and J. Gambetta, "Ibm quantum breaks the 100-qubit processor barrier," *IBM Research Blog*, 2021.

[10] S. M. Clark, D. Lobser, M. C. Revelle, C. G. Yale, D. Bossert, A. D. Burch, M. N. Chow, C. W. Hogle, M. Ivory, J. Pehr, B. Salzbrenner, D. Stick, W. Sweatt, J. M. Wilson, E. Winrow, and P. Maunz, "Engineering the quantum scientific computing open user testbed," *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 1–32, 2021.

[11] I. Cong, S. Choi, and M. D. Lukin, "Quantum convolutional neural networks," *Nature Physics*, vol. 15, no. 12, pp. 1273–1278, Dec. 2019.

[12] S. Debnath, N. M. Linke, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe, "Demonstration of a small programmable quantum computer with atomic qubits," *Nature*, vol. 536, no. 7614, pp. 63–66, Aug. 2016.

[13] S. Ebadi, T. T. Wang, H. Levine, A. Keesling, G. Semeghini, A. Omran, D. Bluvstein, R. Samajdar, H. Pichler, W. W. Ho, S. Choi, S. Sachdev, M. Greiner, V. Vuletić, and M. D. Lukin, "Quantum phases of matter on a 256-atom programmable quantum simulator," *Nature*, vol. 595, no. 7866, pp. 227–232, jul 2021.

[14] M. P. H. et al., "Quantum approximate optimization of non-planar graph problems on a planar superconducting processor," *Nature Physics*, vol. 17, no. 3, pp. 332–336, Feb 2021.

[15] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014.

[16] E. Farhi and A. W. Harrow, "Quantum supremacy through the quantum approximate optimization algorithm," *arXiv: Quantum Physics*, 2016.

[17] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '96. New York, NY, USA: Association for Computing Machinery, 1996, p. 212–219.

[18] J. Kim, T. Chen, J. Whitlow, S. Phiri, B. Bondurant, M. Kuzyk, S. Crain, K. Brown, and J. Kim, "Hardware design of a trapped-ion quantum computer for software-tailored architecture for quantum co-design (staq) project," in *OSA Quantum 2.0 Conference*. Optica Publishing Group, 2020, p. QM6A.2.

[19] I. L. Markov, A. Fatima, S. V. Isakov, and S. Boixo, "Quantum supremacy is both closer and farther than it appears," 2018.

[20] M. Martonosi and M. Roetteler, "Next steps in quantum computing: Computer science's role," *arXiv preprint arXiv:1903.10541*, 2019.

[21] K. K. Mehta, C. Zhang, M. Malinowski, T.-L. Nguyen, M. Stadler, and J. P. Home, "Integrated optical multi-ion quantum logic," *Nature*, vol. 586, no. 7830, p. 533–537, 2020.

[22] N. Moll, P. Barkoutsos, L. S. Bishop, J. M. Chow, A. Cross, D. J. Egger, S. Filipp, A. Fuhrer, J. M. Gambetta, M. Ganzhorn, A. Kandala, A. Mezzacapo, P. Müller, W. Riess, G. Salis, J. Smolin, I. Tavernelli, and K. Temme, "Quantum optimization using variational algorithms on near-term quantum devices," *Quantum Science and Technology*, vol. 3, no. 3, p. 030503, jun 2018.

[23] P. Murali, D. M. Debroy, K. R. Brown, and M. Martonosi, "Architecting noisy intermediate-scale trapped ion quantum computers," in *Proceedings of the ACM/IEEE 47th Annual International Symposium on Computer Architecture*, ser. ISCA '20. IEEE Press, 2020, p. 529–542.

[24] R. J. Niffenegger, J. Stuart, C. Sorace-Agaskar, D. Kharas, S. Bramhavar, C. D. Bruzewicz, W. Loh, R. T. Maxson, R. Mcconnell, D. Reens, and et al., "Integrated multi-wavelength control of an ion qubit," *Nature*, vol. 586, no. 7830, p. 538–542, 2020.

[25] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature Communications*, vol. 5, no. 1, p. 4213, Jul. 2014. [Online]. Available: https://doi.org/10.1038/ncomms5213

[26] I. Pogorelov, T. Feldker, C. D. Marciniak, L. Postler, G. Jacob, O. Krieglsteiner, V. Podlesnic, M. Meth, V. Negnevitsky, M. Stadler, B. Höfer, C. Wächter, K. Lakhmanskiy, R. Blatt, P. Schindler, and T. Monz, "Compact ion-trap quantum computing demonstrator," *PRX Quantum*, vol. 2, p. 020343, Jun 2021.

[27] J. Preskill, "Reliable quantum computers," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1969, pp. 385–410, 1998.

[28] L. J. Stephenson, D. P. Nadlinger, B. C. Nichol, S. An, P. Drmota, T. G. Ballance, K. Thirumalai, J. F. Goodwin, D. M. Lucas, and C. J. Ballance, "High-rate, high-fidelity entanglement of qubits across an elementary quantum network," *Phys. Rev. Lett.*, vol. 124, p. 110501, Mar 2020.

[29] K. Wright, K. Beck, S. Debnath, J. Amini, Y. Nam, N. Grzesiak, J.-S. Chen, N. Pisenti, M. Chmielewski, C. Collins, K. Hudek, J. Mizrahi, J. Wong-Campos, S. Allen, J. Apisdorf, P. Solomon, M. Williams, A. Ducore, A. Blinov, and J. Kim, "Benchmarking an 11-qubit quantum computer," *Nature Communications*, vol. 10, p. 5464, 11 2019.